Deterministic Chaos in Digital Cryptography

by Nikolai V. Ptitsyn

Supervisors:

Professor G. Evans

Professor V. M. Chernenky[†]

submitted in partial fulfillment of the requirements
for the award of a PhD degree at

De Montfort University
The Gateway, Leicester LE1 9BH, United Kingdom
Science and Engineering Research Centre
Institute of Simulation Sciences
Faculty of Computer Science and Engineering

[†]Department of Information Processing and Management Systems
Faculty of Information and Control Systems
Moscow State Technical University (n/a N. E. Bauman), Russia

2003

Abstract

This thesis studies the application of deterministic chaos to digital cryptography. Cryptographic systems such as pseudo-random generators (PRNG), block ciphers and hash functions are regarded as a dynamic system $\langle X, f \rangle$, where X is a state space (i.e. message space) and $f : X \rightarrow X$ is an iterated function. In both chaos theory and cryptography, the object of study is a dynamic system that performs an *iterative nonlinear transformation of information* in an apparently unpredictable but deterministic manner. In terms of chaos theory, the sensitivity to the initial conditions together with the mixing property ensures cryptographic confusion (statistical independence) and diffusion (uniform propagation of plaintext and key randomness into cihertext).

This synergetic relationship between the properties of chaotic and cryptographic systems is considered at both the theoretical and practical levels: The theoretical background upon which this relationship is based, includes discussions on chaos, ergodicity, complexity, randomness, unpredictability and entropy.

Two approaches to the finite-state implementation of chaotic systems (i.e. pseudo-chaos) are considered: (i) floating-point approximation of continuous-state chaos; (ii) binary pseudo-chaos. An overview is given of chaotic systems underpinning cryptographic algorithms along with their strengths and weaknesses. Though all conventional cryposystems are considered binary pseudo-chaos, neither chaos, nor pseudo-chaos are sufficient to guarantee cryptographic strength and security.

A dynamic system is said to have an analytical solution $x_n = \Phi(x_0)$

if any trajectory point x_n can be computed directly from the initial conditions x_0, without performing n iterations. A chaotic system with an analytical solution may have a unpredictable multi-valued map $x_{n+1} = f(x_n)$. Their floating-point approximation is studied in the context of pseudo-random generators.

A cryptographic software system E-Larm™ implementing a multi-stream pseudo-chaotic generator is described. Several pseudo-chaotic systems including the logistic map, sine map, tangent- and logarithm-feedback maps, sawteeth and tent maps are evaluated by means of floating-point computations. Two types of partitioning are used to extract pseudo-random from the floating-point state variable: (i) combining the last significant bits of the floating-point number (for nonlinear maps); and (ii) threshold partitioning (for piecewise linear maps). Multi-round iterations are produced to decrease the bit dependence and increase non-linearity. Relationships between pseudo-chaotic systems are introduced to avoid short cycles (each system influences periodically the states of other systems used in the encryption session).

An evaluation of cryptographic properties of E-Larm is given using graphical plots such as state distributions, phase-space portraits, spectral density Fourier transform, approximated entropy (APEN), cycle length histogram, as well as a variety of statistical tests from the National Institute of Standards and Technology (NIST) suite. Though E-Larm passes all tests recommended by NIST, an approach based on the floating-point approximation of chaos is inefficient in terms of the quality/performance ratio (compared with existing PRNG algorithms). Also no solution is known to control short cycles.

In conclusion, the role of chaos theory in cryptography is identified; disadvantages of floating-point pseudo-chaos are emphasized although binary pseudo-chaos is considered useful for cryptographic applications.

Keywords: chaos, cryptography, synergics, pseudo-chaos, randomness, complexity, unpredictability.

Acknowledgments

First of all, I would like to thank my earlier supervisor, Professor Jonathan Blackledge, representing the British side in the partnership program between De Montfort and Moscow State Technical Universities for his ongoing support that has always been helpful and enjoyable.

I am grateful to my Russian supervisor Professor Valery Chernenky at MSTU who provided both valuable criticism and fresh ideas.

I wish to thank my Professor Gwynne Evans for taking generously my supervising at the end of the project.

I would like to acknowledge the members of Science and Engineering Research Centre at De Montfort University, especially Dr Martin Tuner and Dr Allan Evans for inspiring discussions and advice.

I wish to kindly thank Professor Michael Kapranov and Alexei Tomashevski of the Moscow Power Engineering Institute for their extended feedback.

I am deeply grateful to my parents and family for the variety of their support. My grand farther, Professor at Moscow Power Engineering Institute, contributed to my work giving academic and technical directions.

Finally, this work would not have been possible without research grants provided by Durand Technology Limited.

Notation

X	state space
$X_1, X_2, \cdots$	subsets of X
f	iterated function
x_0	initial condition
p	parameter of an iterated function
f^n	composition of n iterations of f
x_n	system state after n iterations, i.e. $x_n = f^n(x_0)$
$\phi(x_0)$	trajectory starting at x_0
$\Phi(x_0)$	analytical (exact) solution of a dynamic system
s	symbol
α_n	n-symbol string
$K(\alpha)$	algorithmic (Kolomogorov) complexity of α
$c(\alpha)$	symbolic complexity of α
$C(x_0)$	complexity of trajectory staring at x_0
$\Pr(\alpha)$	probability of occurrence of α
$\Pr(\alpha\|\beta)$	probability of α provided β has been obtained
H_n	entropy of α_n
$h_n = h_{n+1\|n}$	conditional entropy of $(n+1)$-th symbol in α_n
h, h_{Sh}	entropy of the source (generator)
h_{KS}	Kolmogorov-Sinai entropy

PAGE MISSING IN ORIGINAL

Glossary of Terms

Cryptography

PRNG	Pseudo-Random Number Generator
LFSR	Linear Feedback Shift Register
NFSR	Nonlinear Feedback Shift Register
LCG	Linear Congruent Generator
SHA1G	Secure Hash Algorithm 1 Generator
S-Box	Substitution Box
P-Box	Permutation Box
PKI	Public Key Infrastructure

Mathematics & Chaos Theory

PDF	Probability Density Function
CDF	Cumulative Distribution Function
KS	Kolmogorov-Sinai (entropy)
APEN	Approximated Entropy
FFT	Fast Fourier Transform

Complexity Theory

P	Polynomial-time Computations
NP	Nondeterministic Polynomial Time Computations
BPP	Bounded-away-error Probabilistic Polynomial Time Computations

Computer Science

CA	Cellular Automata
FPA	Floating-point Arithmetic
BA	Binary Arithmetic
CPU	Central Processing
NaN	Not-a-Number (infinity or incomputable value)

Organizations and Standards

NIST	National Institute of Standards and Technologies
IEEE	Institute of Electronic and Electrical Engineers
FIPS	Federal Information Processing Standard

Contents

Abstract 1

Acknowledgments 3

Notation 4

Glossary of Terms 6

Contents 7

List of Figures 11

1 Introduction 14

1.1 The Information Society 14

1.2 Cryptography, Cryptanalysis and Cryptology 15

1.3 The Synergetic Approach to Cryptography 17

1.4 Deterministic Chaos 19

1.5 History of Chaos-based Cryptography 21

1.6 Original Contribution 23

2 Preliminaries 24

2.1 Cryptography 24

2.1.1 What is Digital Information? 24

2.1.2 Cryptographic System 25

2.1.3 Encryption and Decryption 26

2.1.4 Pseudo-Random Number Generators 32

2.1.5 Confusion and Diffusion 32
2.1.6 Cryptanalysis 33
2.2 Chaotic Dynamics 35
2.2.1 Dynamic Systems 35
2.2.2 Chaotic Systems 36
2.2.3 Attractors 37
2.2.4 Lyapunov Exponents 39
2.2.5 Bifurcation 40
2.2.6 Ergodic and Mixing Systems 40
2.2.7 Binary Chaos 41
2.3 Relationship between Chaos and Cryptography 43

3 Randomness, Complexity and Chaos. 45
3.1 Informal Overview 45
3.2 Complexity Theory Approach 48
3.2.1 Turing Machine 48
3.2.2 Algorithmic Complexity 50
3.2.3 Compressibility and Algorithmic Randomness 50
3.2.4 Symbolic Complexity 51
3.3 Information Theory Approach 51
3.3.1 True Randomness 52
3.3.2 Shannon Entropy 52
3.3.3 Entropy-Complexity Relationship 53
3.4 Entropy and Complexity of Chaotic Systems 54
3.4.1 Partitioning and Symbolic Dynamics 54
3.4.2 Kolmogorov-Sinai Entropy 54
3.4.3 Complexity of the Trajectory 55
3.5 Pseudo-Randomness 56
3.5.1 Probabilistic Ensembles 56
3.5.2 One-Way Function 57
3.5.3 Pseudo-Random Generators 59
3.6 Chaos-based Pseudo-Random Generators 61

4 Pseudo-Chaotic Cryptosystems **65**
4.1 Chaos and Pseudo-Chaos 65
4.2 Floating-point Approximations of Chaos 68
4.2.1 Partitioning the State Space 71
4.2.2 Chebyshev Map 71
4.2.3 Logistic Map 72
4.2.4 Tent Map 76
4.2.5 Other Chaotic Maps 78
4.2.6 Multi-Stream Generators 79
4.2.7 Iteration Counting Ciphers 81
4.2.8 Multi-valued Chaotic Maps with Analytical Solutions 82
4.3 Binary Pseudo-Chaos 84
4.3.1 RANROT 85
4.3.2 Discrete Tent Map 86
4.3.3 Cellular Automata 86
4.3.4 Generalized Baker Map 87
4.3.5 Pseudo-Chaos and Conventional Cryptosystems 89

5 E-Larm: A Chaos Based Encryption System **91**
5.1 Overview 91
5.2 Algorithm 91
5.3 Practical Implementation 95
5.3.1 Platform 95
5.3.2 User Interface 95
5.3.3 Sessions 97
5.4 Initial Conditions and Parameters 100
5.5 Evaluation 101

6 Evaluation of Chaos-based PRNG's **102**
6.1 Introduction 102
6.2 Considerations for Randomness and Unpredictability 106

6.3 Chaotic Maps 106
6.4 Visual Approaches 107
6.4.1 Phase-Space Diagrams 107
6.4.2 Time series 108
6.4.3 Power Spectrum 108
6.4.4 Approximated Entropy 109
6.5 Brief Description of Tests 111
6.6 Performance Evaluation 130
6.7 Conclusions 130

7 Conclusions **132**
7.1 Theory of Chaos based Cryptography 132
7.2 Practical Issues 134
7.3 Future Work 136

Bibliography **138**

Index **149**

List of Figures

1.1 The cross-disciplinary approach to cryptography 18
1.2 Self-organization of the nature 20
1.3 Production of pastry dough [23] 21

2.1 An iterative cryptosystem 25
2.2 A secure communication scheme using encryption and decryption. . . 26
2.3 A trajectory of a block cipher. The block cipher iterates exactly N times. The plaintext p is assigned to the initial state. The ciphertext c is the final state. The key is determined by parameters of the iterated function (not shown on the diagram). 30
2.4 Trajectories of stream ciphers based on a single trajectory. (a) The ciphertext c is the plaintext p plus chaotic noise x (e.g. simple XOR cipher using chaos key stream suggested by Bianco, see Section 4.2.3); (b) The ciphertext c is a number of iterations n (Baptista, Section 4.2.7); (c) The ciphertext c is the system state after p iterations (Gallagher, Section 4.2.7). 30
2.5 A two-dimensional chaotic system: (a) Time space; (b) Phase space. 37
2.6 Binary Chaos: the sensitivity to the initial condition (a) and the topological transitivity (b) are defined on the space Ω. 42
2.7 Phase portraits of chaotic and cryptographic systems.A chaotic system (left) usually has an attractor with a fractal dimension (less than the number of independent variables). A cryptographic system (right) attempts to maximize the dimension up to an integer. 43

3.1 Relationship between the concepts. 46

3.2 Scale of the entropy 47
3.3 A synergy between a chaotic system (top: $\approx$ is a rounding function, $\hat{x_n}$ is the output) and a PRNG (bottom: b is a hard-core predicate, y_n is the output). 60
3.4 A rarefied sample for $k = 6$. 62
3.5 The sawteeth map for $p = 1279$ and $q = 255$. 63

4.1 Properties of chaotic and pseudo-chaotic systems. 66
4.2 Examples of orbits of a pseudo-chaotic system. (a) Dangerously short orbits (unsuitable for cryptography); (b) A single orbit (the best choice for cryptography); (c) Multiple orbits with the same length (also suitable for encryption). 67
4.3 A Laypunov exponent of a chaotic (a) and pseudo-chaotic (b) system 68
4.4 Trajectories of a continuous-state chaotic system (4.4) and its 64-bit floating-point approximation. The first curve is obtained by means of the analytical solution (4.5). The rounding off error is amplified in each iteration and the trajectories diverge exponentially. 69
4.5 The average and the minimal cycle length of the logistic system (4.4) verses floating-point precision obtained from 10 samples of the logistic system. 70
4.6 The empirical Cumulative Distribution Function of a sequence generated with the Chebyshev system before (doted line) and after (solid line) the additional transformation (4.3). The straight line looks like the uniform distribution on $(-2, 2)$. 72
4.7 The logistic map for $r = 0.99$. 73
4.8 A chaotic sequence generated with the logistic map for $x_0 = 0.34$ and $r = 0.99$. 73
4.9 Bifurcation of the logistic map. The most 'unpredictable' behavior occurs when $r \to 1$. 74
4.10 Attractor points corresponding to different values of the parameter r in the Matthews map. 74
4.11 The analytical solution at $n = 5$ of the logistic map for $r = 1$. 75

4.12 The Probability Density Function of a state sequence produced by the logistic system with an incomplete partition. 76
4.13 The tent map. 77
4.14 The Gallagher map. 78
4.15 A multi-stream cryptographic generator 80
4.16 A multiple valued chaotic map $x_{n+1} = f(x_n)$ 83
4.17 A pseudo-random generator based on an analytic solution. 84
4.18 The discrete tent map. 86
4.19 A stretch-and-fold baker transformation. 87
4.20 Six Baker transformations at $\rho = \{0.25, 0.5, 0.25\}$ applied to Lenna [89]. 88
4.21 A typical block cipher is a combination of several pseudo-chaotic systems 89

5.1 E-Larm GUI 97

6.1 Properties of the logistic map 110
6.2 Properties of the logistic map (cont.) 115
6.3 Properties of the sine map 116
6.4 Properties of the tangent-feedback map 117
6.5 Properties of the logarithm-feedback map 118
6.6 Properties of the tent map 119
6.7 Properties of the sawteeth map 120
6.8 Properties of the sawteeth map (cont.) 121
6.9 Properties of the exactly solvable map 122
6.10 Properties of the exactly solvable map (cont.) 123
6.11 Approximated entropy plots 124
6.12 Estimation of *P-value* for `sha-1`, `par`, `par-bad`, `sin` 125
6.13 Estimation of *P-value* for `sha-1`, `par`, `par-bad`, `sin` (cont.) 126
6.14 Estimation of *P-value* for `tan`, `log`, `saw`, `mix` 127
6.15 Estimation of *P-value* for `tan`, `log`,`saw`,`mix` (cont.) 128
6.16 Estimation of *P-value* for `mix` (long sequence) 129
6.17 Performance evaluation of pseudo-random generators 130

Chapter 1

Introduction

1.1 The Information Society

Our digital age has brought about a number of changes in society but perhaps the most profound is the impact it has had upon basic human activities such as decision making, information processing and communications which are all supported by computer devices.

The information revolution can be compared in its impact to that of the industrial revolution of the nineteenth century. The value of information has changed in that both the constructive and destructive powers of information flow are very different compared with the situation less than a hundred years ago. In the commercial sector, 'know how' contributes considerably to company market value because information provides a primary competitive advantage. Clearly, critical information is vital for national security and financial organizations. Accurate knowledge of public opinion allows political leaders to react rapidly in their policies or programs. Thus, political power is now critically dependent on the flow of information.

Organisations in both the public and private sectors have become increasingly dependent on electronic data processing. Vast amounts of digital data are now gathered and stored in large computer data bases and transmitted between computers and terminal devices linked together in complex communications networks. Without appropriate safeguards, these data are susceptible to interception during transmission, or they may be physically removed or copied while in storage. This could result in

unwanted exposures of data and potential invasions of privacy. Data are also susceptible to unauthorised deletion, modification, or addition during transmission or storage. This can result in illicit access to computing resources and services, falsification of personal data or business records, or the conduct of fraudulent transactions, including increases in credit authorisations, modification of funds transfers, and the issue of unauthorised payments.

The speed of information flow has developed considerably over the past ten years. As with high speed vehicles, skilled control becomes increasingly important as information flow increases in the rapidity of its exchange; the web being an environment in which information flow is now very difficult to control effectively.

Access to the global market is a key incentive in society. New economic incentives require new services based on electronic and mobile transactions and both individuals and industries will require involvement under conditions that are not only financially advantageous but secure. The continuous update of security infrastructures is absolutely necessary to encourage the further development of the new global economy.

Modern *information security* manifests itself in many ways according to the situation and requirement. It deals with such concepts as confidentiality, data integrity, access control, identification, authentication and authorization. Practical applications, closely related to information security, are private messaging, financial transactions, online services and many others.

Legislators, recognizing that the importance of information security, have passed laws to help prevent these problems. But laws alone cannot prevent attacks or eliminate threats to data processing systems. Additional steps must be taken to preserve the secrecy and integrity of computer data. Among the security measures that should be considered is cryptography, which embraces methods for rendering data unintelligible to unauthorised parties.

1.2 Cryptography, Cryptanalysis and Cryptology

The word is derived from the Greek *kryptos*, meaning 'hidden', and *graphein* meaning 'to write'. Cryptography concerns the ways in which communications and data can

be encoded to prevent disclosure of their contents through eavesdropping or message interception, using codes, cyphers, and other methods.

From ancient civilisations, with Julius Caesar, Abraham Lincoln and Iosif Stalin cryptography has been a part of history. During the World War II, the Germans developed the Enigma machine to have secure communications. Enigma codes were decrypted first in Poland in the late 1930s and then under the secret 'Ultra Project' based at Bletchly Park in Buckinghamshire (UK) during the early 1940s. This led to a substantial reduction in the level of allied shipping sunk by German U-boats and together the invention of Radar was arguably one of the most important contributions that electronics made to the war effort. In addition, this work contributed significantly to the development of electronic computing after 1945.

Cryptography has always been shrouded in secrecy itself and remains one of the most secret sciences in the world. Professional cryptographers working for intelligence services and commercial organizations have been limited in their publications. As a result, the freely available literature never fully reflects the state of the art. Nations vary in their reticence: whereas the United States released quite generous information on the situation in the Second World War, the Soviet Union clocked itself in silence; but all the time scientists from both countries kept abreast of each other. Claude Shannon, an American electrical engineer, formulated the major criteria and fundamental of cryptographic techniques in his secret report of 1945 that was unclassified in 1949 and published as *Communication theory of secrecy systems* [92]. Vladimir Alexandrovich Kotelnikov, a founder of Russian cryptography, in 1941 formulated the requirements to a perfect encryption system and mathematically proved its cryptographic resistance. This work together with the Kotelnikov theorem (the sampling theorem) became the foundation of Russian cryptography and provided secure communication during World War II in 1942–1945.

Since that time, cryptography has absorbed ideas from complexity theory, number theory, group theory, combinatory logic and modern computer science and grown from basic symmetric ciphers to Public Key Infrastructure (PKI) and complex cryptosystems. In particular, in 1976 asymmetric public keys was first proposed in a revolutionary article by Whitfield Diffie and Martin Hellman [39].

A century ago, one could say that cryptography is the science (or art) of secret

writing and reading, which has grown from semiotics, the study of signs and sign-using behavior[1], rather than mathematics.

Today cryptography is, essentially, the study of mathematical and computational techniques underlying information security. Cryptography is closely connected to the disciplines of cryptanalysis and cryptology. In simple terms, cryptanalysis is the art of breaking cryptosystems, i.e. retrieving the original message without knowing the proper key or forging an electronic signature. Cryptology is the mathematics, such a number theory, which underpin cryptography and cryptanalysis.

Cryptography is the only known practical method for protecting information transmitted through communications networks that uses land lines, communications satellites, and microwave facilities. In some instances, it can be the most economical way to protect stored data. Cryptographic procedures can also be used for message authentication, digital signatures and personal identification for authorising electronic funds transfer and credit card transactions.

1.3 The Synergetic Approach to Cryptography

In making innovations, we typically take existing models from one science and transplant them into a new subject area. For example, it is quite natural to apply models from nonlinear dynamics (chaos theory) for the purpose of encryption. If we are successful, a new cryptographic algorithm will emerge. This is a practical benefit of a cross-disciplinary approach (Figure 1.1).

Unlike most new disciplines that appear at the *edge*, of existing sciences when a model or a technique from one subject area is applied in another, synergetics studies *common fundamentals* of these sciences, extending the global collection of ideas and methods [2].

The term *synergetics* (from Greek: *synergeia* — working together, cooperation) was introduced by the German physicist Haken in the beginning of 70's. Haken's

[1]Although the word was used in this sense in the 17th century by the English philosopher John Locke, the idea of semiotics as an interdisciplinary mode for examining phenomena in different fields emerged only in the late 19th and early 20th centuries.

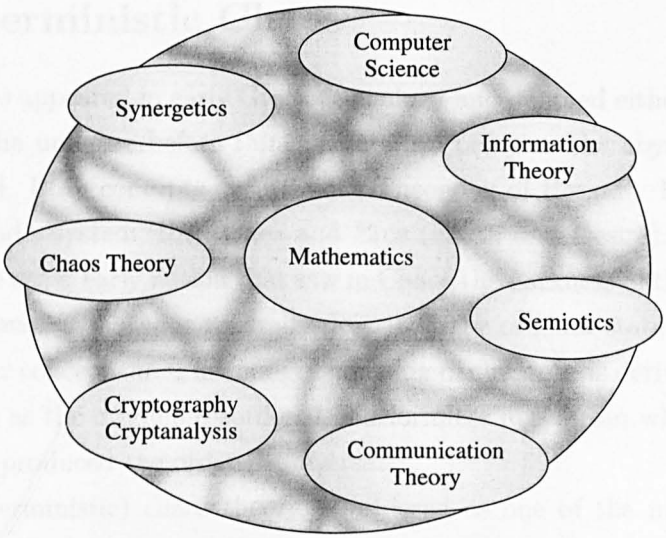


Figure 1.1: The cross-disciplinary approach to cryptography

synergetics treats systems in which cooperation among subsystems creates organized structure on macroscopic scales [54]. Synergetics aims to understand common laws driving processes in different nonlinear systems with a feedback. Examples of problems studied by synergetics are bifurcations, phase transitions in physics, nonlinear oscillations in electrical circuits, population dynamics. From a certain view point, a similiar definition is given by the American scientist R. Buckminster Fuller: "*Synergy means behavior of whole systems unpredicted by the behavior of their parts taken separately*" [47]. One can see this property in cryptographic systems, whose strength depends on the integrity of several mathematical constructions.

Today, it would not be easy to find a discipline remaining untouched by synergetics and deterministic chaos. These new sciences have transfused all the scope of human knowledge: not only mathematics, physics, biology, economics, digital imaging, simulation sciences [20, 6, 18, 4, 16], but also many human studies such as history and sociology [1, 20]. Cryptography, of course, was not an exception — a number of scientists have relatively evaluated encryption techniques based on nonlinear dynamic systems (for example, [76, 53, 69, 45, 63]).

1.4 Deterministic Chaos

The word *chaos* appeared in early Greek cosmology and denoted either the primeval emptiness of the universe before things came into being or the abyss of Tartarus, the underworld. Both concepts occur in the Theogony of Hesiod[2]. First there was Chaos in Hesiod's system, then Gaea and Eros (Earth and Desire). This concept tied in with the other early notion that saw in Chaos the darkness of the underworld.

In later cosmologies, *chaos* generally described the original state of things irrespective of their conception. The modern meaning of the word is derived from Ovid, who saw chaos as the original disordered and formless mass from which the maker of the Cosmos produced the ordered universe.

Today (deterministic) chaos theory, considered as one of the most important sciences of the last few decades, has absorbed ideas and methods from pure mathematics, dynamical systems theory, symbolic dynamics, non-equilibrium thermodynamics and fractal geometry. Compared with Greek cosmology, it denotes the opposite properties of chaos: self-organization, order, bounded space and determinism. However, both meanings of the word assume the *unpredictability.*

The idea that many simple nonlinear deterministic systems can behave in an apparently unpredictable and chaotic manner was first observed by the great French mathematician Henri Poincaré. Other early pioneering works in the field of chaotic dynamics are to be found in the mathematical literature by such luminaries as Birkhoff, Cartwright, Littlewood, Levinson, Smale, Kolmogorov and his students.

The key feature of chaotic behavior in different systems is mainly related to the high sensitivity to initial conditions due to exponential divergence of all trajectories lying on the attracting structure which is normally bounded in an appropriate phase space. In the 1960's Edward Lorenz, an American meteorologist, discovered a stable chaotic attractor and predicted that: *"... it may happen that small differences in the initial conditions produce very great ones in the final phenomena. A small error in the former will produce an enormous error in the future. Prediction becomes impossible..."*

The Nobel laureate Ilia Prigogine, a Russian-born Belgian physical chemist, is

[2]Hesiod, 700 BC, one of the earliest Greek poets. His epic *Theogony* describes the myths of the gods.

Figure 1.2: Self-organization of the nature

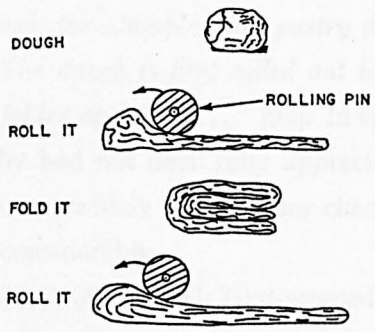


Figure 1.3: Production of pastry dough [23]

one of the founders of chaos theory. His research in nonlinear dissipative structures led to the concept of equilibrium and non-equilibrium conditions to categorize the state of a system. In the physical studies of thermodynamics, Prigogines' research revealed non-equilibrium conditions that led to systemic behavior different from what was expected by the customary interpretation of the second law of thermodynamics. The phenomena of bifurcation and self-organization emerged from systems in equilibrium as if there was disruption or interference. This disruption or interference became the next step in chaos theory; it became *chaos/complexity* theory.

Mathematical or deterministic chaos is a dynamic system, characterized with a 'complex' and 'unpredictable' behavior. Intuitively, this property suits the requirements of a digital encryption system — on the one hand computer-based cryptosystems are deterministic; on the other, they must be cryptographically unpredictable. Practically, the last property implies that given certain information on the ciphertext and the plaintext (the message), a cryptanalyst should not be able to predict the cryptographic transformation and recover the key or the message.

1.5 History of Chaos-based Cryptography

In his classical paper Claude Shannon explicitly mentioned the basic stretch-and-fold mechanism of chaos for the purpose of encryption (Figure 1.3): *"Good mixing transformations are often formed by repeated products of two simple noncommuting*

operations. Hopf[3] *has shown, for example, that pastry dough can be mixed by such a sequence of operations. The dough is first rolled out into a thin slab, then folded over, then rolled, and then folded again, etc...* " [92]. In spite of this, the importance of chaos-based cryptography had not been fully appreciated till the end of 1980's when digital computers became widely available for chaos simulations and the role of cryptography increased considerably.

Many papers (e.g. [67, 33, 3, 36, 104, 93, 7]) discussed encryption schemes based on synchronized chaotic circuits. However, these *analog* schemes belong more to the field of steganography and secure radio communication, and lie beyond the scope of this thesis.

Cryptography attracted scientists from many areas and they started exploiting different dynamic systems for the purpose of encryption. Most of these systems were defined on real numbers, so floating-point arithmetic was used to approximate chaos on a finite-state machine (digital computer) [41, 59, 76, 25, 53, 82, 49, 22, 103, 69].

Discrete (binary) chaotic systems [97] such as the cellular automata [102, 52], discrete Baker transformation [89, 45], discrete tent map [75] and discrete affine transformation [84] provided more efficient encryption schemes.

Since 2000, the potential of chaos-based communication, especially spread spectrum modulation, has been recognized worldwide. For instance, the book *Chaotic electronics in telecommunications* by Kennedy *et al.* [63] described chaotic modulations and suggested their electronic implementations. Again, the emphasis here is put on *information coding* rather then *digital encryption*, which is the subject of this research.

Finally, inspiration for this work came from my supervisor J. M. Blackledge [26], L. Kocarev [66], Z. Kotulski and J. Szczepański [68, 95].

[3]Hopf, Eberhard F. F, (1902–1983), an Austrian mathematician who made significant contributions in topology and ergodic theory, studied 'pastry dough mixing' in compact spaces [*On causality, statistics and probability*, Journal of Mathematics and Physics, 13, pp. 51–102, 1934]

1.6 Original Contribution

This thesis discusses the theoretical background and practical implementations of chaos-based cryptosystems known at the present moment. Additionally, the following results are considered useful and unique:

- Floating-point approximations of various chaotic systems (i.e. pseudo-chaos) are studied in deep using different types of partitioning methods and statistical tests.
- Known cryptographic techniques using chaos have been combined and extended into an original system E-Larm. In particular,
  - i multiple nonlinear chaotic systems are used together to increase the reliability and complexity of the generator;
  - ii a mixing component is used to provide a relationship between different chaotic systems and increase the average cycle length of the generator in general;
  - iii the minimal bound of the iteration counter (discrete time) is used to improve the pseudo-random appearance of the output; whereas the upper bound is used to avoid very short orbits;
  - iv optimal values for algorithm parameters are derived and implemented.
- An evaluation of the E-Larm system is given using a test methodology based on NIST (National Institute of Standards and Technologies) recommendations.
- Conclusions are made about the general role of chaos theory in cryptography.

Chapter 2

Preliminaries

This chapter assembles the mathematical concepts from cryptography [90, 77] and nonlinear dynamics [81, 11, 12, 9, 4, 31, 79], emphasizing their common fundamentals.

2.1 Cryptography

2.1.1 What is Digital Information?

With regard to information in a computer environment we assume an array of bits or binary strings that carry a text message or any multimedia object such as an image and sound. A binary string to be transmitted or kept confidential in a storage is called *plaintext*. We denote the plaintext as

$$p = \{p_1, p_2, \ldots, p_i, \ldots, p_n | \, p_i \in \mathcal{P}\},$$

where $\mathcal{P}$ is a finite alphabet of plaintext symbols. In digital cryptosystems, the alphabet is binary, that is $\mathcal{P} = \mathbb{Z}_2 = \{0, 1\}$. We can also consider the alphabet of bytes ($\mathcal{P} = \mathbb{Z}_{256} = \{0, 1\}^8$) or larger data blocks.

Let $c = \{c_1, c_2, \ldots, c_i, \ldots, c_n | \, c_i \in \mathcal{C}\}$ denote the encrypted message or the *ciphertext*. The alphabet $\mathcal{C}$ can differ from $\mathcal{P}$, in most cases, $\mathcal{P} = \mathcal{C} = \{0, 1\}^m$.

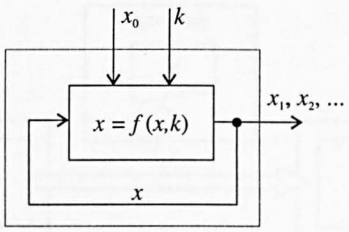


Figure 2.1: An iterative cryptosystem

2.1.2 Cryptographic System

In a broad sense, a cryptographic system is the whole computer infrastructure providing information security, i.e. a balanced software and hardware solution implementing cryptographic algorithms (encryption/decryption, authentication, key management etc). In the real world a cryptographic system crucially depends on humans.

In a narrow sense, a cryptographic system is a mathematical construction that transforms information to make it *inaccessible* for an adversary. A cryptosystem can be considered as a dynamic system (Section 2.2.1)

$$\mathcal{S} = \langle X, f : X \times \mathcal{K} \to X \rangle , \tag{2.1}$$

where the set X is a state space and the function f is a cryptographic transformation depending on a key from $\mathcal{K}$. The system state $x \in X$ encodes certain information, for example, a plaintext or a signature (in most ciphers, $X = \mathcal{P} = \mathcal{C}$). Other examples of cryptographic systems are a pseudo-random generator and hash functions. The function f is given by a deterministic algorithm, which can be implemented on a Turing machine.

Often, a cryptosystem is defined by a *nonlinear iterative transformation with a feedback* (Figure 2.1). The system produces a sequence of states $x_0, x_1, \ldots, x_n, \ldots$, where $x_n = f(x_{n-1}, k) = f^n(x_0, k)$, $x_0 \in X$, $k \in \mathcal{K}$. The initial condition x_0 is an input message, and the final state x_n is an output message. The sequence $\{x_n\}$ is called a *trajectory* or an *orbit*. Since f is deterministic, the whole trajectory is given by the initial condition x_0 and the (secret) parameter k.

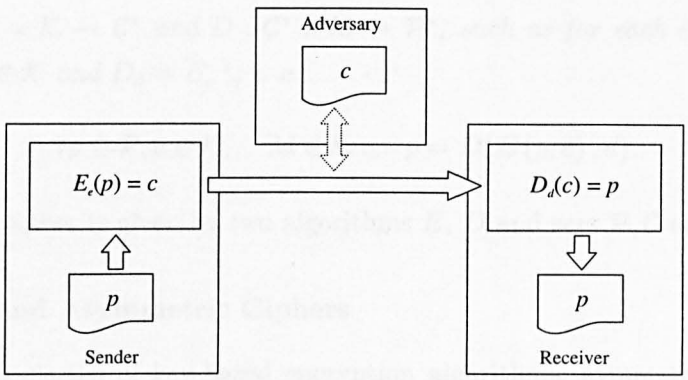


Figure 2.2: A secure communication scheme using encryption and decryption.

So far, any conventional cryptographic system can be considered as a deterministic dynamic system $\langle f, X, \mathcal{K} \rangle$ with a nonlinear iterated function f, a state space X and a key space $\mathcal{K}$. In general, the basic requirements for a cryptosystem are unpredictability and random-like behavior.

2.1.3 Encryption and Decryption

Encryption is a transformation $E : \mathcal{P}^* \times \mathcal{E} \to \mathcal{C}^*$, where $\mathcal{C}$ is a ciphertext alphabet, and $\mathcal{E}$ is an encryption key space. Hence,

$$c = E(p, e) = E_e(p), \quad p \in \mathcal{P}, c \in \mathcal{C}, e \in \mathcal{E}.$$

Decryption is the inverse transformation $D : \mathcal{C}^* \times \mathcal{D} \to \mathcal{P}^*$, where $\mathcal{D}$ is a decryption key space, i. e.

$$p = D(c, d) = D_d(c), \quad p \in \mathcal{P}, c \in \mathcal{C}, d \in \mathcal{D}.$$

Usually, $\mathcal{E} = \mathcal{D} = \mathcal{K} = \{0, 1\}^m$.

Figure 2.2 gives a classical communication scheme using encryption and decryption.

Definition 1 *Encryption scheme or cipher is a system*

$$\mathcal{S} = \langle E, D, \mathcal{P}, \mathcal{C}, \mathcal{K} \rangle,$$

where $E : \mathcal{P}^* \times \mathcal{K} \rightarrow \mathcal{C}^*$ *and* $D : \mathcal{C}^* \times \mathcal{K} \rightarrow \mathcal{P}^*$, *such as for each* $e \in \mathcal{K}$ *exists a unique key* $d \in \mathcal{K}$ *and* $D_d = E_e^{-1}$, *i. e.*

$$\forall p \in \mathcal{P}, e \in \mathcal{K}, \quad \exists d \in \mathcal{K} : \quad p = D\big(E\left(p, e\right), d\big).$$

Practically, a cipher is given by two algorithms E, D and sets $\mathcal{P}, \mathcal{C}$ and $\mathcal{K}$.

Symmetric and Asymmetric Ciphers

There are two classes of key-based encryption algorithms, symmetric (or private key) and asymmetric (or public key) algorithms. The difference is that *symmetric algorithms* use the same key for encryption and decryption ($e = d$, or the decryption key is easily derived from the encryption key), whereas *asymmetric algorithms* use a different key for encryption and decryption ($e \neq d$), and the decryption key cannot be derived from the encryption key.

Asymmetric ciphers (also called public-key algorithms or generally public-key cryptography) permit the encryption key to be public (it can even be published in a newspaper), allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the public key and the decryption key the private or secret key. A well known example of asymmetric cipher is cryptographic system developed in 1977 by Rivest, Shamir and Adleman (RSA) in the US [87]. Pretty Good Privacy (PGP), an encryption software written by Philip R. Zimmerman, is an other example of public key cryptography [106].

This paper focuses on symmetric ciphers.

Substitution and Transposition Ciphers

Before the development of digital computers, cryptography consisted of character-based algorithms. Different cryptographic algorithms either substituted characters for one another or transposed characters with one another. The better algorithms did both, many times each.

Although the technology for developing cypher systems is more complex now, the underlying philosophy remains the same. The primary change is that algorithms

work on bits instead of characters. This is actually just a change in the alphabet size—from 26 elements to 2^{256} elements in modern block ciphers.

Substitution Cyphers As their name suggests, these preserve the order of the plaintext symbols, but disguise them. Each letter or group of letters is replaced by another letter or group to disguise it. In its simplest form, 'a' becomes 'D', 'b' becomes 'E', 'c' becomes 'F' etc.

More complex substitutions can be devised, e.g. a random (or key controlled) mapping of one letter to another. This general system is called a monoalphabetic substitution. They are relatively easy to decode if the statistical properties of natural languages are used. For example, in English, 'e' is the most common letter followed by 't', then 'a' etc.

The cryptanalyst would count the relative occurrences of the letter in the cyphertext, or look for a word that would be expected in the message. To make the encryption more secure, a polyalphabetic cypher may be used, in which a matrix of alphabets is employed to smooth out the frequencies of the cyphertext letters.

It is in fact possible to construct an unbreakable cypher if the key is longer than the plaintext, although this method, known as a 'one time pad' has practical disadvantages.

Transposition Cyphers A common example, the 'column transposition cypher' is shown in Table 2.1 Here the plaintext is: 'This is an example of a simple transposition cypher'. The cyphertext is: 'almniefheolpnatnepsorimsripdspiathesaatsicixfeocb'

The plaintext is ordered in rows under the key which numbers the columns so formed. Column 1 in the example is under the key letter closest to the start of the alphabet. The cyphertext is then read out by columns, starting with the column whose number is the lowest.

To break such a cypher, the cryptanalyst must guess the length of the keyword, and order of the columns.

K	E	Y	W	O	R	D
3	2	7	6	4	5	1
t	h	i	s	i	s	a
n	e	x	a	m	p	l
e	o	f	a	s	i	m
p	l	e	t	r	a	n
s	p	o	s	i	t	i
o	n	c	i	p	h	e
r	a	b	c	d	e	f

Table 2.1: Example of Transposition Cypher

Block and Stream Ciphers

Encryption algorithms can also be divided into stream ciphers and block ciphers. *Stream ciphers* (also known as state ciphers) encrypt a single bit or block of plaintext at a time and the encryption depends on time (i.e. cipher state); identical plaintext blocks are transformed into different ciphertext blocks. *Block ciphers* take a larger block of bits and encrypt them as independent units; identical plaintext blocks correspond to identical ciphertext blocks. For example, the transposition cipher is a typical block encryption scheme.

A stream or block cipher can be considered as a nonlinear dynamic system (2.1). The role of trajectory in each class of ciphers is explained below:

1. A block cipher performs n iterations with an *invertible* function f (Figure 2.3). The number n is fixed and relatively small (often, $n = 16$). Each iteration transfers the system into the next state, i. e. $x_{i+1} = f(x_i)$. The plaintext is assign to the initial condition ($x_0 = p$), and the ciphertext is obtained from the final state ($c = x_n$). The intermediate states $(x_1, \ldots, x_{n-1})$ are hidden from external observers. Symbolically,

$$E(p,k) = f^n(p,k),$$
$$D(c,k) = f^{-n}(c,k).$$

Modern block ciphers combine several dynamic systems by adding their states

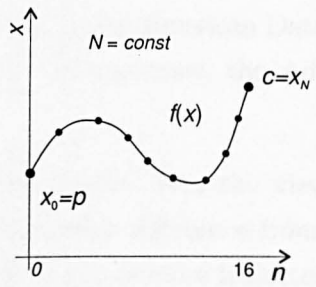


Figure 2.3: A trajectory of a block cipher. The block cipher iterates exactly N times. The plaintext p is assigned to the initial state. The ciphertext c is the final state. The key is determined by parameters of the iterated function (not shown on the diagram).

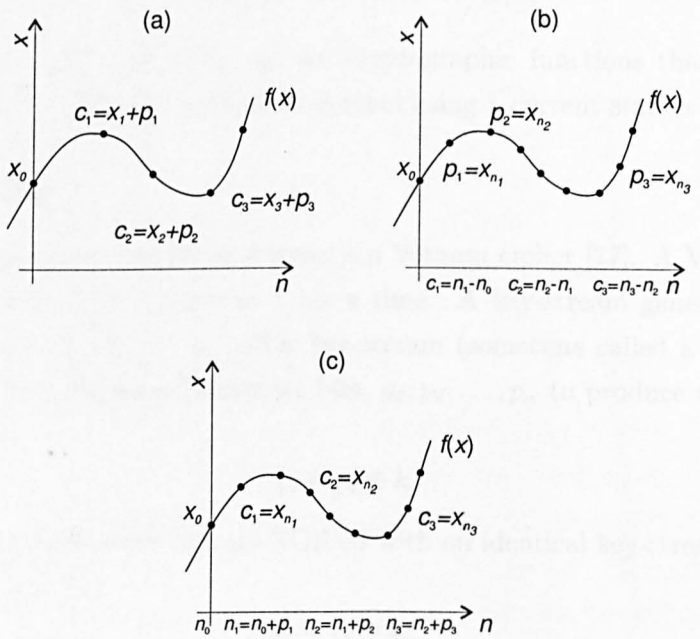


Figure 2.4: Trajectories of stream ciphers based on a single trajectory. (a) The ciphertext c is the plaintext p plus chaotic noise x (e.g. simple XOR cipher using chaos key stream suggested by Bianco, see Section 4.2.3); (b) The ciphertext c is a number of iterations n (Baptista, Section 4.2.7); (c) The ciphertext c is the system state after p iterations (Gallagher, Section 4.2.7).

to each other, for example, in the American Data Encryption Standard (DES) developed at IBM or in its successor, the Advanced Encryption Standard (AES).

2. Stream ciphers are more diverse from the view point of using the trajectory (Figure 2.4 a-c). The main difference from block ciphers is that all the ciphertext is 'cohered' with one or more trajectories and the encryption transformation *each block* depends on the system state x. The trajectory length is not limited and depends on the message length. Chapter 4 provides example ciphers for each type *a,b* and *c*. Simplifying, we can write

$$E(p_n, k) = g(f^n(x_n, k), p_n),$$
$$D(c_n, k) = g^{-1}(f^n(x_n, k), c_n)$$

where $g(x, p_n)$ and $g^{-1}(x, c_n)$ are cryptographic functions that respectively encrypts and decrypts the input symbol using a current state x as the key.

Vernam Cipher

The simplest stream encryption scheme is a Vernam cipher [77]. A Vernam cipher converts plaintext into ciphertext 1 bit a time. A key-stream generator outputs a stream of bits: $k_1, k_2, \ldots, k_n$. This key-stream (sometime called a running key) is XOR'ed with a stream of plaintext bits, $p_1, p_2, \ldots, p_n$ to produce the stream of cipher bits, i.e.

$$c_i = p_i \oplus k_i$$

To decrypt, the ciphertext bits are XOR'ed with an identical key-stream to recover the plaintext bits, i.e.

$$p_i = c_i \oplus k_i,$$

since $p_i \oplus k_i \oplus k_i = p_i$.

Figure 2.4 (a) represents a similar encryption scheme. The trajectory corresponds to the key-stream. After each iteration $c_i = p_i + x_i$ (for decryption $p_i = c_i - x_i$).

If the key-stream $k = \{k_i\}$ is truly random (has a truly uniform distribution), the Vernam cipher is called one-time pad, which is perfectly secure. The concept of true randomness and perfect security is discussed in Chapter 3.

2.1.4 Pseudo-Random Number Generators

In practical cryptography, one-time pads are not used since it is technically difficult to generate and distribute enormous truly random keys. Random pads are not reproducible. Pseudo-Random (Number) Generators (PRNG) extract a compact seed into a long sequence. The output pseudo-random sequences appear random, i. e. the probability distribution function cannot be efficiently distinguish from that of white noise.

PRNG's play a central role in computer cryptography. Although they may be applied explicitly in stream ciphers, the main application is round key generation in block encryption schemes.

According to our approach, we consider PRNG's as a nonlinear dynamic system that differs from the conventional approach in which pseudo-random numbers are generated from mod-based iterations (e.g. $x_{n+1} = rx_n \bmod q$ as illustrated in Section 3.6). Figure 2.1 represents a typical PRNG producing a sequence $x_1, x_2, \ldots$ from the seed given by the initial condition x_0 and parameter p.

The critical requirements for dynamic systems to be used in cryptography are pseudo-randomness and unpredictability. Chapter 3 provides formal definitions of these properties and links them to concepts of chaotic dynamics.

2.1.5 Confusion and Diffusion

Using pseudo-random sequences, modern algorithms are far from the theoretical perfection and thus retain some information about the plaintext in the ciphertext. Roughly speaking, the randomness or unpredictability of the seed is spread through the whole pseudo-random sequence. An amount of plaintext redundancy is still invariant after encryption (kept in the ciphertext), which makes possible both known-plaintext and ciphertext-only attacks. To avoid this (at least to some degree), the plaintext should be reduced as close as possible to its true entropy by means of a good compression algorithm. If ideal compression could be achieved, then changing any number of bits in the compressed message would result in another sensible message when uncompressed (see also Section 3.2.3). Otherwise, two basic techniques for hiding redundancies in ciphertext should be used (Shannon, [92]):

Confusion

Confusion ensures that the (statistical) properties of plaintext blocks are not reflected in the corresponding ciphertext blocks. Instead every ciphertext must have a pseudo-random appearance to any observer or standard statistical test.

Diffusion

In terms of *plaintexts*, diffusion demands that (statistically) similar plaintexts do result in completely different ciphertexts even when encrypted with the same key. In particular, this requires that any element of the input block influences every element of the output block in a complex irregular fashion.

In terms of a *key*, diffusion demands that similar keys do result in completely different ciphertexts even when used for encrypting the same block of plaintext. This requires that any element of the key influences every element of the output block in a complex irregular fashion. Additionally, this property must also be valid for the decryption process because otherwise an intruder might recover parts of the input block from an observed output by a partly correct guess of the key used for encryption.

Most block ciphers explicitly implement confusion and diffusion: respectively, their iterated function consists of substitution and permutation phases. Eventually, both phases taken together, ensure a highly nonlinear transformation resulting in a pseudo-random and unpredictable ciphertext.

2.1.6 Cryptanalysis

The whole point of cryptography is to keep the plaintext (or the key, or both) secret from eavesdroppers (also called adversaries, attackers, interceptors, interlopers, intruders, opponents, or simply the enemy). Eavesdroppers are assumed to have complete access to the communication between the sender and receiver.

Cryptanalysis is the science of recovering the plaintext of a message without access to the key. Successful cryptanalysis may recover the plaintext or the key. It also may find weaknesses in a cryptographic system that eventually leads to recovery of the plaintext or key. (The loss of a key though non-cryptanalytic means is called a compromise.)

A fundamental assumption in cryptanalysis (first enunciated by the Dutchman A Kerckhoff) assumes that the cryptanalyst has complete details of the cryptographic algorithm and implementation. Some old cryptographic systems rely on the secrecy of the algorithms; such algorithms are only of historical interest and are not adequate for real-world needs. All modern algorithms use a key to control encryption and decryption ("*The enemy knows the system being used*" as stated by Shannon); a message can be decrypted only if the key matches the encryption key.

An attempted cryptanalysis is called an attack. There a four principal types of cryptanalytic attacks; each of them assumes that the cryptanalyst has complete knowledge of the encryption algorithm used:

Cyphertext-only attack The cryptanalyst has the cyphertext of several messages, all of which have been encrypted using the same encryption algorithm. The cryptanalyst's job is to recover the plaintext of as many messages as possible, or to deduce the key (or keys) used to encrypt the messages, in order to decrypt other messages encrypted with the same keys.

Known-plaintext attack The cryptanalyst not only has access to the cyphertext of several messages, but also to the plaintext of those messages. The problem is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key (or keys).

Chosen-plaintext attack The cryptanalyst not only has access to the cyphertext and associated plaintext for several messages, but also chooses the plaintext that gets encrypted. This is more powerful than a known-plaintext attack, because the cryptanalyst can choose specific plaintext blocks to encrypt those that might yield more information about the key. The problem is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key (or keys).

Adaptive-chosen-plaintext attack This is a special case of a chosen-plaintext attack. Not only can the cryptanalyst choose the plaintext that is encrypted, but can also modify the choice based on the results of previous encryption. In a chosen-plaintext attack, a cryptanalyst might just be able to choose one large

block of plaintext to be encrypted; in an adaptive-chosen-plaintext attack it is possible to choose a smaller block of plaintext and then choose another based on the results of the first, and so on.

In addition to the above, there are at least three other types of cryptanalytic attack.

Chosen-cyphertext attack The cryptanalyst can choose different cypher-texts to be decrypted and has access to the decrypted plaintext. For example, the cryptanalyst has access to a tamperproof box that does automatic decryption. The problem is to deduce the key. This attack is primarily applicable to public-key algorithms. A chosen-cyphertext attack is sometimes effective against a symmetric algorithm as well. (A chosen-plaintext attack and a chosen-cyphertext attack are together known as a chosen-text attack).

Chosen-key attack This attack does not mean that the cryptanalyst can choose the key; it means that there is some knowledge about the relationship between different keys - it is a rather obscure attack and not very practical.

2.2 Chaotic Dynamics

2.2.1 Dynamic Systems

A dynamic *continuous-state continuous-time* system $\mathcal{S} = \langle X, \mathcal{K}, f \rangle$ depending on parameters can be considered in the following form

$$\frac{dx}{dt} = f(x, k), \quad x \in X \subseteq R^d, k \in \mathcal{K} \subseteq R^{d_\mathcal{K}},$$

where $f : X \times \mathcal{K} \to Y$ is a smooth vector function, X is a state space, and $\mathcal{K}$ is a parameter space. The equation satisfies the conditions of the existence and uniqueness of solutions $x(t, x_0)$ with the initial condition $x_0 = x(0, x_0)$ [4, 11]. The solution curve $x(t, x_0)$, is called the trajectory.

In cryptography, we focus on a dynamic *discrete-time* system, which can be written in the following form:

$$x_{n+1} = f(x_n, k), \quad x_n \in X \subseteq R^d, k \in \mathcal{K} \subseteq R^{d_\mathcal{K}}, n = 0, 1, 2, \ldots \tag{2.2}$$

where x_i is a discrete state of the system. The trajectory $\varphi(i, x_0)$ is defined by the sequence $x_0, x_1, x_2, \ldots$ Clearly, equation (2.2) is similar to a cryptographic iterated function, used in PRNG, block ciphers and other constructions (see Figures 2.1–2.4). Consequently, in both nonlinear dynamics and cryptography we deal with *an iterated key-dependent transformation of information.*

2.2.2 Chaotic Systems

There are several sufficient conditions satisfied by a dynamic system to guarantee chaos [29]; the sensitivity to initial conditions and topological transitivity are the most common.

Definition 2 (chaotic dynamic system, [31, 79, 29, 4]) *A chaotic continuous-state discrete-time system (continuous chaos for short) is a dynamic system* $\mathcal{S} = \langle X, f \rangle$ *with two properties:*

1. *Given a metric space* $X \subseteq R^N$ *and a mapping* $f : X \to X$, *we say that* f *is* topological transitive *on* X *if for any two open sets* $U, V \subset X$, *there is* $n \geq 0$ *such that* $f^n(U) \cap V \neq \emptyset$.

2. *The map* f *is said to be* sensitive to initial conditions *if there is* $\delta > 0, n \geq 0$ *such that for any* $x \in X$ *and for any neighborhood* H_x *of* x *there is* $y \in H_x$, *such that* $|f^n(x) - f^n(y)| > \delta$.

This can be interpreted in natural language as follows: *a dynamic system is chaotic if all trajectories are bounded and nearby trajectories diverge exponentially at every point of the phase space* (see Figure 2.5[1]). A synergy between chaotic and cryptographic systems section 2.1 can be described as follows:

[1]Trajectories illustrated in Figures 2.5 are *continuous* and belong to a *two dimensional* system said to be chaotic. This diagram should be interpreted only as an example to be extended in more dimensions. Clearly, a 2D dynamic system with continuous trajectories cannot be chaotic (otherwise any interception of trajectories will lead to periodical cycles) unlike 3 or more dimensional system. This note should not be applied to discrete-time system (2.2) where chaotic behavior is possible in 1 or 2 dimensions.

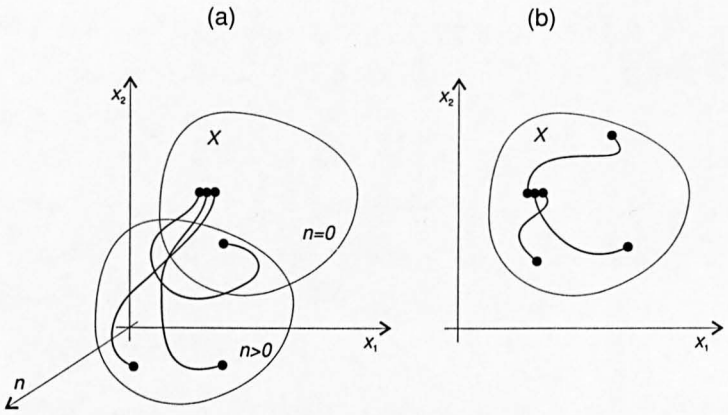


Figure 2.5: A two-dimensional chaotic system: (a) Time space; (b) Phase space.

The topological transitivity ensures that the system output covers all the state space e.g. any plaintext can be encrypted into any ciphertext.

The sensitivity to the initial condition corresponds to Shannon's requirements for an encryption system (section 2.1.5). In both chaos and cryptography we are dealing with systems in which "*a small variation of any variable changes the outputs considerably*" [92].

2.2.3 Attractors

Most nonlinear systems are unpredictable and yet patterned, it is called strange and since it tends to produce a fractal[2] geometric shape. It is said to be attracted to that shape. Tsonis defines attractor as "*a limit set that collects trajectories*" [96]. When we observe the dynamic behaviour of the state point in the state space we can notice that trajectories tend to contract in certain areas; this contraction is called *the attractor*. The attractor is actually *a set of points such that all trajectories*

[2]The term *fractal* was coined by Benoit Mandelbrot in 1975. It comes from the Latin *fractus*, meaning an irregular surface like that of a broken stone. Fractals are non-regular geometric shapes that have the same degree of non-regularity on all scales. "*Fractal Geometry plays two roles. It is the geometry of deterministic chaos and it can also describe the geometry of mountains, clouds and galaxies*"—Benoit Mandelbrot.

nearby converge to it. At the same time, by definition, nearby trajectories diverge exponentially at every point. The coexistence of these two properties is another reason for the 'strange name' of the attractor.

Chaos is centered on the concept of the strange attractor. If we watch the flow of water from a faucet we turn, so the water goes faster and faster; we will see activity from smooth delivery to gushing states. These various kinds of flow represent different patterns to which the flow is attracted.

The four kinds of attractors are:

1. Point attractor, such as a pendulum swinging back and forth and eventually stopping at a point. The attractor may come as a point, in which case, it gives a steady state where no change is made.

2. Periodic attractor, just add a mainspring to the pendulum to compensate for friction and the pendulum now has a limited cycle in its phase space. The periodic attractor portrays processes that repeat themselves.

3. Torus attractor, picture walking on a large doughnut, going over, under and around its outside surface area, circling, but never repeating exactly the same path you went before. The torus attractor depicts processes that stay in a confined area but wander from place to place in that area.

4. Strange attractor, this attractor deals with the three-body problem of stability. The strange attractor shows processes that are stable, confined and yet never do the same thing twice. Three non-linear equation solutions exhibit a fractal structure in computer simulations of the strange attractor. In other words, each solution curve tended to the same area, the attractor area, and cycled around randomly without any particular set number of times, never crossing itself, staying in the same phase space, and displaying self-similarity at any scale. The operative term here is self-similarity. Each event, each process, each period, each end-state in phase-space is never precisely identical to another; it is similar but not identical.The attractor acts on the system as a whole and collects the trajectories of perturbation in the environment. Though these systems are unstable, they have patterned order and boundary.

The term 'attractor' has no counterpart in cryptography, precisely, cryptography attempts to hide the attractor and any other recognizable *structure* of the system, making it *unpredictable.*

2.2.4 Lyapunov Exponents

Definition 2 introduced the concept of the sensitivity to the initial condition. The Lyapunov exponent, denoted by $\lambda(x_0)$, is a quantitative measure of the exponential divergence of trajectories starting from the neighborhood of x_0 [16]. For a one-dimensional system

$$|f^n(x_0+\varepsilon) - f^n(x_0)| = \varepsilon \cdot e^{n\lambda(x_0)},$$

where ε is a small perturbation from the initial condition x_0 and n is the number of iterations (the discrete time). Generally, λ depends on the initial condition, so we can estimate its average value. In a measure-preserving system (2.2.6) λ is constant for all trajectories. The Lyapunov exponent is given by the limit

$$\lambda(x_0) = \lim_{n\to\infty}\lim_{\varepsilon\to 0}\frac{1}{n}\log\left|\frac{f^n(x_0+\varepsilon) - f^n(x_0)}{\varepsilon}\right| \tag{2.3}$$

or

$$\lambda(x_0) = \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\log|f'(x_k)| = \lim_{n\to\infty}\frac{1}{n}\log\prod_{k=1}^{n}|f'(x_k)| \tag{2.4}$$

For each k, $f'(x_k)$ tells us how much the function f is changing with respect to its argument at the point x_k. This derivative expresses the magnitude of change in the transition from x_k to x_{k+1}. The limit of the average of the derivative logarithms over n iterations is taken to provide a measure of how fast the orbit changes as (discrete) time propagates. A positive Lyapunov exponent is an indication of chaotic behavior.

For a d-dimensional system we have a set $\lambda = \{\lambda_1, \ldots, \lambda_d\}$ and more complex behavior, but still qualitatively the same as the one dimensional case [78].

A more accurate measure is the Kolmogorov-Sinai (KS) entropy because it considers the resolution (the precision) under which the system is observed. Chapter 3 studies the complexity of chaotic systems using the KS entropy.

From the view point of cryptography, λ measures the encryption efficiency. Larger λ indicates that fewer cryptography transformation will be required to reach a

certain level of the ciphertext unpredictability. Chapter 4 provides numeric values of λ for certain chaos-based encryption systems.

2.2.5 Bifurcation

The term bifurcation (from Latin *bifurcus* - two pronged) usually means a qualitative transformation from regular to chaotic behavior as a control parameter is smoothly varied. There are several types of bifurcation; the best known is period doubling discovered by Feighenbaum (see Figure 4.9). In bifurcation points, the number of equilibrium states doubles. As the control parameter increases, this happens more often, finally, when the number of possible states is infinitely large, the system becomes chaotic.

Certain cryptographic applications use control parameters as a key. Clearly, the key space should correspond to chaotic behavior. In a good encryption system, the number of states and the length of orbits should not depend on the choice of parameters.

2.2.6 Ergodic and Mixing Systems

Let the dynamic system $\mathcal{S} = \langle X, f \rangle$ have a f-invariant measure μ [8], $\mu(X) < \infty$, that is

$$\forall A \in \sigma(X), \qquad \mu(A) = \mu\left(f^{-1}(A)\right),$$

where $\sigma(X)$ is the σ-algebra of measurable subsets of X. Assume the f-invariant measure is equivalent to the Lebesgue with the density function $g(x)$ bounded with some positive constants g_1 and g_2:

$$0 < g_1 < g(x) < g_2,$$

where $\forall A \in \sigma(X), \quad \mu(A) = \int_A g(x)\,dx$. If g_1 is close to g_2 then the measure μ is close to the uniform distribution.

A dynamic system $\mathcal{S} = \langle X, f \rangle$ is ergodic if it has only trivial invariant sets, i.e. if and only if either $\mu(A) = 0$ or $\mu(X \setminus A) = 0$, whenever A is a measurable, invariant under f, subset of the space X (the invariance of A means $f(A) \subset A$). Ergodicity implies that the space X cannot be divided into invariant nontrivial (with respect to

the measure μ) disjoint parts (in the case of smaller disjoint parts, any brute force attack is restricted to one part of the partition i.e. an intruder will have to search through the whole state space X) [95].

A dynamical system $\mathcal{S} = \langle X, f \rangle$ is mixing if it satisfies the condition

$$\forall C, P \in \sigma(X), \qquad \lim_{n \to \infty} \left(\frac{\mu\left(f^{-n}(C) \cap P\right)}{\mu(P)} \right) = \frac{\mu(C)}{\mu(X)}.$$

If $\mu(X) = 1$ (the measure μ is probabilistic) we can write

$$\lim_{n \to \infty} \left(\mu\left(f^{-n}(C) \cap P\right) \right) = \mu(C)\,\mu(P).$$

The mixing property implies that the part of P, which after n iterations of f is contained in C is asymptotically proportional to the volume (in the sense of the measure μ) of C in X. Moreover, the iterations of f make each set C statistically independent from P (asymptotically). In other words, the trajectory starting at a fixed point $x_0 \in X$, after n iterations reaches any region with the same probability. Vice versa, for a fixed final state x_n and sufficiently large n, any state x_0 is μ-equiprobable [68] (see also section 3.6).

2.2.7 Binary Chaos

We have discussed the main properties of chaos defined on real numbers. However, digital cryptography is based on discrete state systems.

In 1998 Waelbroeck and Zertuche [97] proposed a theory of deterministic chaos for binary systems. Binary systems, like a cellular automata and neural networks, are the best known class of chaotic discrete systems. Consider a state space

$$\Omega = \{\alpha | \alpha \in \mathbb{Z}_2^*\}, \quad \mathbb{Z}_2 = \{0, 1\}.$$

The system state $\alpha \in \Omega$ is given by a sequence of symbols (bits):

$$\alpha = \alpha(1)\alpha(2)\ldots\alpha(n)\ldots$$

A natural topological measure for binary systems is the Hamming distance defined as

$$d_H(\alpha, \alpha') \equiv \sum_{i=1}^{n} |\alpha(i) - \alpha'(i)|,$$

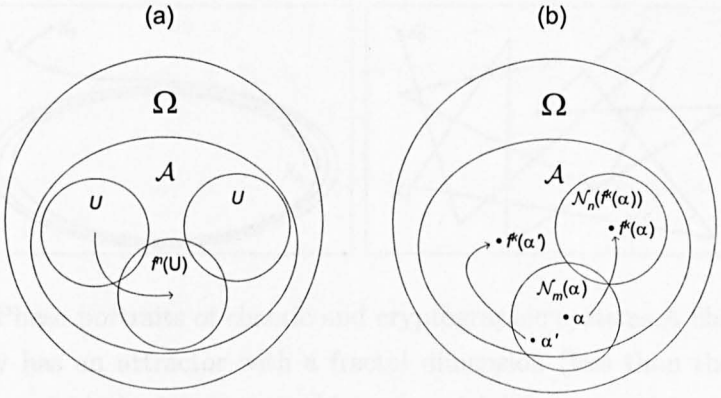


Figure 2.6: Binary Chaos: the sensitivity to the initial condition (a) and the topological transitivity (b) are defined on the space Ω.

i. e. d_H is a number of bits by which the two strings α and α' differ. However, in the limit $n \to \infty$ (infinite strings) the Hamming distance cannot define the topology of Ω. For this case we can introduce a topological base

$$\mathcal{N}_n(\alpha) = \{\alpha' \in \Omega | \alpha(i) = \alpha'(i), \forall i < n\},$$

where $n = 1, 2, 3 \ldots$. Now we can generalize the definition of chaos from R^d to Ω:

Definition 3 (binary chaos, [97]) Let $\mathcal{A} \subset \Omega$ and $f : \mathcal{A} \to \mathcal{A}$. The binary system $\langle \mathcal{A}, f \rangle$ is said chaotic, if two following conditions are satisfied:

1. The function f is *topological transitive* on $\mathcal{A}$, i. e. for all open subsets $U, V \subset \mathcal{A}$ it exists $n \in \mathbb{Z}$ such as $f^n(U) \cap V \neq \emptyset$ (Figure 2.6-b)[3].

2. The function f is *sensitive to the initial conditions* on $\mathcal{A}$, i. e. for all $\alpha \in \mathcal{A}$ and $\mathcal{N}_m(\alpha)$ it exists $n, k \in \mathbb{N}$ and $\alpha' \in \big(\mathcal{N}_m(\alpha) \cap \mathcal{A}\big)$, such as $f^k(\alpha') \notin \mathcal{N}_n\big(f^k(\alpha)\big)$ (Figure 2.6-a).

Hence, a chaotic binary system is defined in a similar way as real space chaos. Section 4.3 discusses the Lyapunov exponent for finite state systems, and Sections 3.6 and 4.3.5 show that conventual cryptographic schemes can be considered as 'limited' binary chaos.

[3] If $n < 0$ and f is not invertible, we assume $f^n(U) = \{\alpha \in \mathcal{A} | f^{-n}(\alpha) \in U\}$.

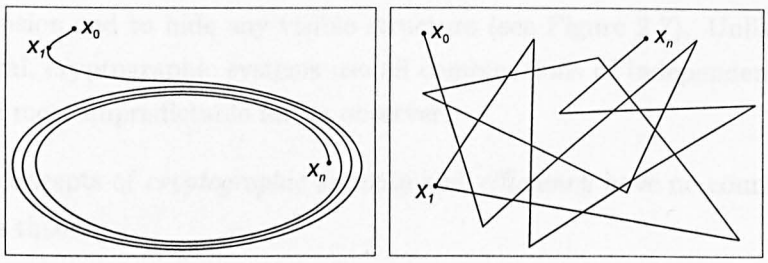


Figure 2.7: Phase portraits of chaotic and cryptographic systems.A chaotic system (left) usually has an attractor with a fractal dimension (less than the number of independent variables). A cryptographic system (right) attempts to maximize the dimension up to an integer.

2.3 Relationship between Chaos and Cryptography

So far, we have considered cryptographic systems from the view point of synergetics and chaotic dynamics. Clearly, chaos theory and cryptography studies a *nonlinear iterative transformation* depending on parameters. Such properties as sensitivity to initial conditions, exponential divergence of trajectories and mixing can be applied in both areas.

In terms of the object of study the following differences are important:

1. Chaos theory studies nonlinear systems defined on an *infinite state* space (e.g vectors of real numbers or infinite binary strings), whereas cryptography relies on a *finite-state machine* (computer). All chaos models implemented on a computer are approximations, i. e. *pseudo-chaos.*

2. Chaos theory studies the *asymptotic behavior* of a nonlinear system ($n \to \infty$), whereas cryptography focuses on the effect of a *small number of iterations* ($n \ll \infty$). For example, $n = 16$ in a block cipher and $n \sim$ `[message size]` in a stream cipher.

3. Classical chaotic systems have visually *recognizable attractors* (the phase space portrait of the system) whose dimension is fractional[4]. Cryptography attempts to maximize the attractor dimension to an integer, the topological

[4]We say a set is n-dimensional if we need n independent variables to exactly identify any state

dimension,and to hide any visible structure (see Figure 2.7). Unlike chaos in general, cryptographic systems use all combinations of independent variables to be most unpredictable for an observer.

4. The concepts of *cryptographic security* and *efficiency* have no counterparts in chaos theory.

The following table summarizes the relationship between the concepts.

Chaos theory	**Cryptography**
chaotic system	pseudo-chaotic system
— nonlinear transformation	— nonlinear transformation
— infinite number of states	— finite number of states
— infinite number of iterations	— finite number of iterations
initial condition	plaintext
final state	ciphertext
initial conditions and parameters	key
asymptotic independence of the initial and final states	confusion
the sensitivity to the initial condition and parameters, mixing	diffusion

It is natural to suggest using known chaotic systems for the purpose of encryption. Section 3.6 provides a theoretical construction of a chaos-based PRNG and Chapter 4 is devoted to its computer implementations (pseudo-chaos).

of the system. This notion of dimension is called the *topological dimension* of a set. Fractal objects, in particular, chaotic attractors, can have less degrees of freedom than its topological dimension because not all combinations of independent variables are possible. For example, a geometrical fractal plotted on a 2D plane might have the dimension greater than 1 (corresponding to line) and less than 2 (corresponding to surface).

Chapter 3

Randomness, Complexity and Chaos.

The concepts of randomness, unpredictability, complexity and entropy form the basis of modern cryptography. Mathematically, the design of a cryptosystem can be interpreted as the design of a key-dependent bijective transformation, most unpredictable for an observer with a given amount of resources.

The present chapter links these concepts with chaotic dynamics and suggests a pseudo-random generator based on a chaotic system.

3.1 Informal Overview

In studying a cryptosystem (PRNG, encryption algorithm or key exchange scheme), a cryptanalyst has access to time series of this dynamic system and knows the iterated function (the algorithm). However, the time series is not a compact subset of the trajectory (intermediate states are hidden) and the iterated function has a secret parameter (the key). We can think of the sample as being 'random', 'unpredictable' and 'complex'. What do these properties mean mathematically? How do they relate to chaos? This chapter attempts to give an answer.

Perfect security is achieved when the cryptosystem is **absolutely unpredictable** for an external observer, i.e. all possible outcomes (states, sub-trajectories) are equiprobable and do not depend on the previous states. In other words, the state

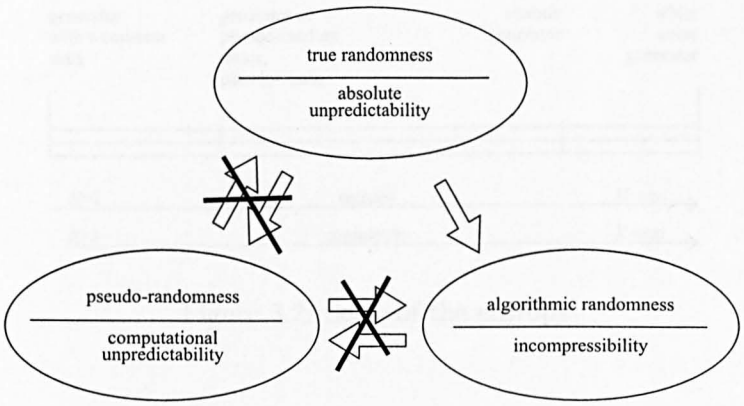


Figure 3.1: Relationship between the concepts.

sequence has a uniform probability distribution and no correlations (patterns). The concept of absolute unpredictability is equivalent to the **true randomness** and related to the **white noise**. A source of the white noise can be a highly dimensional system (e. g. microscopic motion in the ideal gas).

In the real world, cryptographic systems provide a certain level of **practical security** that is much 'lower' that the perfect security (due to usability and cost reasons). At this level we deal with 'pseudo' concepts. **Pseudo-random** sequences cannot be efficiently distinguished from the uniform noise by known algorithms. Similarly, computationally unpredictable sequences cannot be predicted with the available computer resources. It is possible to show that the pseudo-random sequence (cryptosystem) is computationally unpredictable, and vice versa.

Together with probabilistic properties we shall consider the **algorithmic complexity**, i.e. the length of the shortest algorithm producing the cryptographic sequence. Intuitively, the (internal) complexity of the system provides the (external) unpredictability. A sequence is called **algorithmically random**, if its algorithmic complexity equals the length of the sequence. An algorithmically random sequence is **computationally incompressible** and contains no recognizable patterns (redundancies).

Clearly, a purely random system is also algorithmically random (Figure 3.1). The concepts of pseudo and algorithmic randomness are different: a pseudo-random

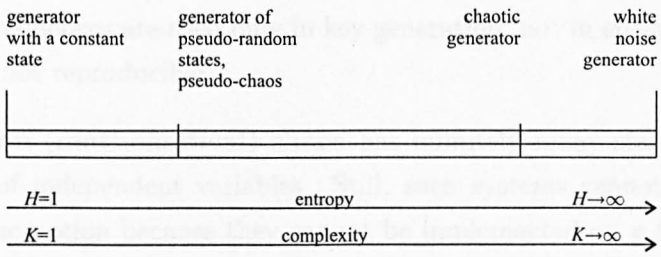


Figure 3.2: Scale of the entropy

string is generated with a compact seed, but the external observer is not able (practically) to reconstruct the generator and predict the sequence. In other words, the string is highly compressible for authorized communication parties, but computationally incompressible for the adversary. In the general case, an algorithmic random string can be predicted by a probabilistic machine.

Randomness or unpredictability can be 'measured' using such properties as the algorithmic complexity or the entropy i.e. the degree of our uncertainty about the system. Quantitatively, the Shannon entropy is in direct proportion to the algorithmic complexity (in ergodic systems, where statistical properties of a single sequence coincides with that of all sequences, emitted by a generator). A randomness measure for chaos is the Kolmogorov-Sinai entropy that is, roughly speaking, a multiresolution integration of Lyapunov exponents.

A 'fully predictable' system has a well-know state, its complexity is 1. A fully unpredictable system (the uncorrelated white noise) is an infinite source of information and infinite complexity and entropy (Figure 3.2). Cryptographic systems are somewhere in-between: they are complex enough to be unpredictable by an external observer, but not too much to be reproducible.

The following types of chaos can be considered in the context of cryptography:

Natural chaos (e.g. the environment) is a highly dimensional system with infinitely many states and independent variables. Generally, the entropy of such a system is not maximal because of its self-organization and correlations. There are cryptographic applications using natural chaos; for example, Intel's hardware RNG captures randomness from the thermal noise of the computer. Such

random sequences are used only in key generation, not in encryption, because they are not reproducible.

Low-dimension (mathematical) chaos has infinitely many states and a small number of independent variables. Still, such systems cannot be applied to digital encryption because they cannot be implemented on a finite-state machine. This chapter discusses the cryptographic properties of mathematical chaos and studies a chaos-based PRNG with infinite states. In the following chapter we attempt to transform the results into different types of approximations (pseudo-chaos).

3.2 Complexity Theory Approach

3.2.1 Turing Machine

Throughout this thesis we will use the common terminology from complexity theory [5, 73]. For the sake of self-sufficiency we provide a brief introduction to the subject.

A Turing machine is a hypothetical device that can theoretically implement any computer algorithm. It provides a unified framework to measure the complexity (i.e. program length and working time) of computational problems such as generating, transforming and matching cryptographic sequences.

A basic model of a Turing machine consists of (1) a semi-infinite tape divided into cells, (2) a read-write head (3) a control unit, which is a finite-state sequential machine. We denote a Turing machine as

$$T = \langle S, \mathcal{A}, \Gamma, F, q_0 \rangle ,$$

where S is the finite state set of the control; $\mathcal{A}$ is the finite tape alphabet ($\mathcal{A} = \{0, 1\}$); Γ is a finite rule state of the form $\gamma : S \times \mathcal{A} \rightarrow S \times \mathcal{A} \times \{L, N, R\}$; $F \subseteq S$ is a set of halting accepting states; and $q_0 \in S$ is the initial state. The state $\{L, N, R\}$ includes tape commands: 'move left' (L), 'stay in place' (N) and 'move right' (P). *The machine configuration* T is the triplet $\langle s, \alpha, i \rangle$, where $s \in S$ is the current state, $\alpha \in \mathcal{A}^*$ the tape string, and $1 \leq i \leq |\alpha|$ is the head position counting from the left end of the tape.

The machine is initialized in the following way: (1) a string $\alpha \subset \mathcal{A}^*$ is loaded onto the tape; (2) the head is seeded to the leftmost position; and (3) the initial state q is assigned to the state variable s. In every step, (1) the machine reads a symbol from the current cell, depending on the symbol read and the current state, (2) it makes a transition to a new state; (2) it overwrites the current cell with a new symbol; (3) it moves the head one step left or right, or stays in place. The machine halts when a state $f \in F$ is reached.

A Turing machine is said to *accept* a string α if a sequences of rules $\gamma_1, \ldots, \gamma_m \subset \Gamma^*$ exists that puts the machine from the initial state s_0 to any halting accepting state $f \in F$. The machine *rejects* a string, if it halts in $s \notin F$ or if it never halts.

Turing machines may take several other forms: (a) The tape may be extended to make it doubly infinite; (b) One or more additional tapes may be added; (c) The storage medium may be the plane ruled into squares or n-dimensional space ruled into n-dimensional cubes. It can be shown that each of these extensions of the basic machine are equivalent in the sense that a basic machine can be constructed that 'simulates' the behavior of the extended machine.

A *language* $\mathcal{L}$ over a finite alphabet $\mathcal{A}$ is a subset $(A)^*$, i.e. a subset of all finite strings over $\mathcal{A}$. A machine is said to accept a language $\mathcal{L}$ if it accepts all the strings $\alpha \in \mathcal{L}$ and rejects $\beta \notin \mathcal{L}$.

A *deterministic Turing machine* has single-valued rules $\gamma : S \times \mathcal{A} \rightarrow S \times \mathcal{A} \times \{L, N, R\}$, i.e. there are no rules with the same left parts $S \times \mathcal{A}$. By contrast, *nondeterministic machines* may have multi-valued rules.

If there exists a polynomial $p(l)$, limiting the machine working time m (the number of steps) depending of the input string length l ($m < p(l)$), the machine is said to run in polynomial time. The complexity class **P** is the set of languages accepted by deterministic polynomial-time machines. The complexity class **NP** is the set of languages accepted by nondeterministic polynomial-time machines.

A *Probabilistic Turing machine* is a deterministic Turing machine that can flip a fair coin to determine its next move. A probabilistic machine is said to accept a language $\mathcal{L}$ if it enters an accepting state for $\alpha \in \mathcal{L}$ with the probability $p_1 > 2/3$ and it halts $\alpha \notin \mathcal{L}$ with the probability $p_0 > 2/3$.

The complexity class **BPP** consists of all languages recognized by probabilistic

polynomial-time machines.

3.2.2 Algorithmic Complexity

The concept of algorithmic complexity was suggested independently by three mathematicians A. N. Kolmogorov, P. Solomonov and G. J. Chaitin:

Definition 4 *The algorithmic complexity $K_M(\alpha)$ of a finite string $\alpha \in \{0,1\}^n$ with respect to a Turing Machine M is the length $l(\pi)$ of the smallest computer program π, which generate it, that is*

$$K_M(\alpha) = \min_{\pi:M(\pi)=\alpha} l(\pi).$$

Kolmogorov showed, that there exists a *universal* Turing machine U, that performs computations equivalent to π (designed for an arbitrary machine M), and the changes of π required to adopt it for U depend on M but not on α. Consequently, the algorithmic complexity K_M with respect to any machine M is related to $K_U(S)$ by

$$K_U(S) \leq K_A(S) + C_A, \tag{3.1}$$

where C_M is a constant, which is independent from α [21]. Hereafter we omit the subscript U assuming $K(\alpha) = K_U(\alpha)$.

Unfortunately, algorithmic complexity cannot be commuted i.e. there is no universal solution for simplifying programs and for proving that the length is minimal. We cannot apply this definition directly to compare the complexity of cryptographic sequences or algorithms. Nevertheless the theoretical applications are very important. In particular, the Kolmogorov complexity provides a unified approach to the problem of data compressibility.

3.2.3 Compressibility and Algorithmic Randomness

A string α_n of length n is said to be c-incompressible if $K(\alpha_n) \geq n-c$. Incompressible strings ($c = 0$ or relatively small) are called algorithmically random.

3.2.4 Symbolic Complexity

For an infinite string α_∞ or a generator, it is interesting to consider the symbolic complexity given by the limit

$$c(\alpha_\infty) = \lim_{n\to\infty} \frac{K(\alpha_n)}{n}. \tag{3.2}$$

From (3.1) it follows that the symbolic complexity $c(\alpha_\infty)$ is invariant with respect to the choice of Turing machine. If a string has a finite Kolmogorov complexity (e. g. a pseudo-random string), its symbolic complexity tends to 0. A truly random string has $c = 1$ because its length equals the length of the shortest program. Clearly, $c > 0$ if and only if the generator has infinite complexity. In chaotic systems, this happens if the complexity of the initial conditions is infinitely large or a certain amount of randomness gets into the system from the environment.

3.3 Information Theory Approach

A cryptanalyst considers the cryptosystem as a source of information. In studying statistical properties of the ciphertext, he/she attempts to reconstruct the secret transformation (precisely, secret parameters of a known algorithm). In ideal cryptosystems, the distribution of the ciphertext cannot be differentiated from the uniform noise, and thus provides no useful information for an adversary.

The two major approaches to statistical analysis of cryptographic strings are as follows:

- Studying properties of one particular string leads to complexity theory as introduced in Section 3.2.
- By contrast, information theory founded by Shannon [91] describes the properties of all strings produced by the source (cryptosystem).

In ergodic systems (in which statistical properties of a particular string coincides with that of all strings emitted by a generator) both approach can be unified (see section 3.3.3).

3.3.1 True Randomness

We define a Probability Distribution Function (PDF) as a function from strings $\mathcal{L} = \{\alpha_j\}$ to nonnegative real numbers, i.e. $\Pr : \mathcal{L} \to [0,1]$ such that $\sum_{\alpha \in \mathcal{L}} \Pr(\alpha) = 1$.

Definition 5 *A string α is called truly (purely) random (or unpredictable) if for any substrings $\beta_n, \gamma_n \in \alpha$, $0 < n < length(\alpha)$*

$$\Pr(\beta_n) = \Pr(\gamma_n)$$

.

A truly random string cannot be predicted, i.e. for any symbol $s_i \in \alpha$, the conditional probability $\Pr(s_i | s_{i-1}, s_{i-2}, \ldots) = \Pr(s_i)$. In other words, an arbitrary large knowledge about the previous states does not increase the probability of the successful prediction of the next state.

An infinite and truly random string has a delta autocorrelation function, and an infinite and uniform power spectrum (white noise).

3.3.2 Shannon Entropy

Claude Shannon generalized the term *entropy* from thermodynamics to abstract problems in communication and information theories [91]. The entropy measures the amount of information required to determine precisely the system state among all possible states. In other words, it is a relative degree of our uncertainty or the lack of information we have about the exact state of a system. In cryptography, the entropy is related the unpredictability of an encryption system for an adversary.

The entropy of a string α_n of the length n is defined as

$$H_n = -\sum_{\alpha \in \mathcal{A}^n} \Pr(\alpha_n) \log_{|\mathcal{A}|} \Pr(\alpha_n), \tag{3.3}$$

where $\Pr : \mathcal{A}^n \to [0,1]$ is the PDF of α_n on the set of n-th symbol strings. The maximum of H_n is achieved when $\Pr(\alpha_n)$ is a uniform distribution (the string is truly random).

The conditional entropy h_n denotes the average amount of information supplied with each $(n+1)$-th symbol provided the previous n symbols are known:

$$h_n = h_{n+1|n} = \begin{cases} H_{n+1} - H_n, & n \geq 1 \\ H_1, & n = 1 \end{cases}$$

In other words, h_n quantifies the average uncertainty when predicting the next symbol. As soon as the knowledge about a previous state cannot increase the uncertainty, the function H_n is non decreasing and $h_{n+1} \leq h_n$.

For a stationary information source there exists a limit

$$h_{Sh} = \lim_{n\to\infty} h_n = \lim_{n\to\infty} \frac{H_n}{n}, \tag{3.4}$$

called the entropy of information source (cryptographic system).

If α is a k-th order Markov sequence, $h_n = h_{Sh}$ for all $n \geq k$. A Markov sequence corresponds to a deterministic process, in which the next state depends on the previous k states, that is

$$\Pr(s_i|s_{i-1}, s_{i-2}, \ldots) = \Pr(s_i|s_{i-1}, s_{i-2}, \ldots, s_{i-k}), \quad s_i \in \alpha.$$

Examples of Markov process can be found in most cryptographic systems such as PRNG's and block ciphers.

3.3.3 Entropy-Complexity Relationship

Intuitively, complexity and the entropy are related as 'cause and effect': the more complex the internal organization of a system, the more unpredictable its behavior is and the higher the entropy is. The complexity is the size of the 'internal program' that generates a state sequence (string), whereas the entropy is computed from the probability distribution of this sequence. Formally, the following result is applied to stationary ergodic sources [28]:

$$\lim_{n\to\infty} \frac{\langle K_n \rangle}{H_n} = \frac{1}{\ln 2}, \tag{3.5}$$

where $\langle K_n \rangle = \sum_{\alpha_n \in \{0,1\}^n} \Pr(\alpha_n) K(\alpha_n)$. Hence, the average complexity $\langle K_n \rangle$ is asymptotically proportional (with the coefficient $\ln 2$) to the entropy as n increases.

3.4 Entropy and Complexity of Chaotic Systems

3.4.1 Partitioning and Symbolic Dynamics

Consider a chaotic system $\mathcal{S} = \langle X, f \rangle$ with an f-invariant measure μ (see Section 2.2). Any set of m disjoint regions which covers the state space X is a partition denoted by

$$\beta = \{X_1, X_2, ..., X_m\} : \quad \bigcup_{i=1}^{i=m} X_i = X, \quad X_i \cap X_j = \emptyset, \ \forall i \neq j.$$

A unique symbol $s_i \in \mathcal{A}$ is assigned to every region X_i. The process of partitioning the state space, assigning symbols to every region from the partition, and the resulting macroscopic dynamics are called *symbolic dynamics* [55]. A function σ can define partitions and their symbolic names:

$$\sigma(x) = \{s_i \in \mathcal{A} | x \in X_i\}.$$

A trajectory $\phi(x_0)$ passing across the subsets X_i produces a *symbolic trajectory* $\alpha(x_0)$.

3.4.2 Kolmogorov-Sinai Entropy

The Lyapunov exponents (Section 2.2.4) measure how fast we loose the capability to predict the behavior of a chaotic system in time. The disadvantage is that this measure does not consider the resolution under which the system is observed, unlike the Kolmogorov-Sinai entropy [16, 9, 28].

Let the partition $\beta = \{X_1, X_2, ..., X_m\}$ be the observer's resolution. Looking at the system state x, the observer can only determine the fact that $x \in X_i$ and reconstruct the symbolic trajectory $\alpha_n = \{s_{m_1}, s_{m_2}, \ldots, s_{m_n}\}$ corresponding to the regions visited. The entropy of a trajectory α_n with the respect to partition β is given by

$$H_n^\beta = -\sum_{\alpha_n} \Pr(\alpha_n) \log_{|\mathcal{A}|} \Pr(\alpha_n),$$

where $\Pr(\alpha_n)$ is the probability of occurrence of the substring α_n. The conditional entropy of the $(n+1)$-th symbol provided the previous n symbols are known is

defined as

$$h_n^\beta = h_{n+1|n}^\beta = \begin{cases} H_{n+1}^\beta - H_n^\beta, & n \geq 1 \\ H_1^\beta, & n = 1 \end{cases}$$

The entropy for a partition β is given by

$$h^\beta = \lim_{n\to\infty} h_n^\beta = \lim_{n\to\infty} \frac{1}{n} H_n^\beta.$$

The Kolmogorov-Sinai entropy of a chaotic system is the supremum over all possible partitions

$$h_{KS} = \sup_\beta h^\beta. \tag{3.6}$$

The KS entropy equals zero for regular systems, is finite and positive for a deterministic chaos and infinite for a random process. It is related to the Lyapunov exponents by $h_{KS} = \sum_{1\leq d\leq D} \lambda_d$ and proportional to the time horizon T on which the system is predictable.

3.4.3 Complexity of the Trajectory

The complexity of the trajectory of a point x_0 with respect to a finite opening coverings β is defined as

$$\mathcal{C}^\beta(x_0) = \limsup_{n\to\infty} \frac{1}{n} \min_{\alpha_n \in [\psi(x)]^n} K(\alpha_n),$$

where $[\psi(x)]^n = \{\alpha_n | f^j(x_0) \in X_j\}$ and $K(\alpha_n)$ is the algorithmic complexity of α.

The complexity of the trajectory of a point x_0 is

$$\mathcal{C}(x_0) = \sup_\beta \mathcal{C}^\beta(x_0).$$

Definition 6 (algorithmically random trajectory, [30, 101]) *The trajectory of a point x_0 is called algorithmically random if its complexity is positive, that is $c(x_0) > 0$.*

The Brudno and White theorem defines the relationship between the KS entropy and complexity:

Theorem 1 (complexity of the trajectory, [30, 101]) *The symbolic trajectories of almost all $x \in X$ (with respect to the invariant measure μ) are algorithmically random and their complexity is given by*

$$c(x) = \frac{h_{KS}}{\ln 2}, \tag{3.7}$$

Though it is practically impossible to quantify the algorithmic complexity of a string, most strings over a finite alphabet produced by a chaotic system are algorithmically random.

3.5 Pseudo-Randomness

3.5.1 Probabilistic Ensembles

Let $\Pr_i(\alpha)$ be a probability distribution function of strings $\{0,1\}^{l(i)}$, where $l(i)$ is a positive polynomial. We write $\Pi = \{\Pr_i\}_{i\in I}$ for an ensemble of distributions indexed by $I \subset \mathbb{N}$. The ensemble of the uniform distributions $\Pi_0 = \{\Pr_{0,i}\}_{i\in\mathbb{N}}$ for all $i \in \mathbb{N}$ and $\alpha, \beta \in \{0,1\}^i$ satisfies $\Pr_{0,i}(\alpha) = \Pr_{0,i}(\beta)$.

To measure the 'degree of randomness' of a string its probability ensemble should be compared with that of the uniform distributions. Having limited resources, computers can process only a subset of distributions. Thus, we introduce the concept of polynomial indistinguishability. Roughly speaking, two probabilistic ensembles are polynomially indistinguishable if they assign 'about the same' mass to the same subsets of strings, efficiently recognized by a Turing machine:

Definition 7 (polynomial indistinguishability, [105, 50, 65]) *Let $\Pi_1 = \{\Pr_{1,i}\}_{i\in I}$ and $\Pi_2 = \{\Pr_{2,i}\}_{i\in I}$ be two probability ensembles each indexed by I. Let T be a probabilistic polynomial-time Turing machine called a test. The test gets two inputs: an index i and a string α. Let $\Pr_1^T(i)$ be the probability that, on input index i and a string α chosen according to the distribution $\Pr_{1,i}$, the test T outputs 1. Similarly, $\Pr_2^T(i)$ denotes the probability that, on input index i and a string α chosen according to the distribution $\Pr_{2,i}$, the test T outputs 1. We say that Π_1 and Π_2 are* indistinguishable *with polynomial $p(i)$ if for all probabilistic polynomial-time tests T and sufficiently large $i \in \mathbb{N}$*

$$\left|\Pr_1^T(i) - \Pr_2^T(i)\right| < \frac{1}{p(i)}.$$

Definition 8 (pseudo-random probability ensemble, [105, 50, 65]) *The probability ensemble* $\Pi = \{\Pr_i\}_{i\in I}$ *is said to be pseudo-random if for any positive polynomial* $p(i)$*, the ensemble* Π *is indistinguishable with* $p(i)$ *from the uniform ensemble* $\Pi_0 = \{\Pr_{0,i}\}_{i\in I}$.

Definition 9 (unpredictable probability ensemble, [105, 50, 65]) *Let* $\Pi = \{\Pr_{1,i}\}_{i\in I}$ *be a probabilistic ensemble indexed by* I*. Let* T *be a probabilistic ensemble polynomial-time Turing machine that on input (index* i *and a string* α*), outputs a single bit, called the guess. Let* $bit(\alpha, r)$ *denote the* r*-th bit of the sequence* α*, and* $pref(\alpha, r)$ *denote the prefix of* r *bits of the string* α*, (i.e.* $pref(\alpha, r) = bit(\alpha, 1)\, bit(\alpha, 2) \ldots bit(\alpha, r)$*). We say that the machine* T *predicts the next bit of* Π*, if for some polynomial* $p(i)$ *and infinitely many* i*'s,*

$$\Pr\left(M\left(i, pref(\alpha, r)\right) = bit(\alpha, r+1)\right) \geq \frac{1}{2} + \frac{1}{p(i)},$$

where the probability space is that of the string α *chosen according to* $\Pr_{1,i}$*, and the integer* r *chosen at random with uniform distribution in* $\{0, 1, \ldots, l(\alpha) - 1\}$*. We say that* Π *is unpredictable if there exists no probabilistic polynomial time machine* T *which predicts the next bit of* Π.

Theorem 2 [27, 50, 65] *The probability ensemble* Π *is pseudo-random if and only if* Π *is unpredictable.*

3.5.2 One-Way Function

One-way functions are functions that are easy to evaluate ($\beta = f(\alpha)$), but hard (on the average) to invert ($\alpha = f^{-1}(\beta)$). The computational gap between forward and inverse evaluation quantifies the efficiency of the one-way transformation. This is related to the so-called inverse problems including inverse scattering theory for example, for which exact solutions are rare in practice, the problems ill-posed in most cases.

It is widely believed, but not proven that one-way functions exists. However, one-way functions lie at the heart of modern cryptography, in particular, they are used in public-key schemes.

A function is called length-preserving (denoted by 1:1) if the binary length of its argument α equals the length of the result β. An important application of the one-way 1:1 function is the PRNG.

If a one-way function is not length-preserving [$length(\alpha) > length(\beta)$], it is called a hash-function. Hash functions are used, for example, in digital signature schemes.

A formal definition of one-way function is given in terms of complexity theory:

Definition 10 (one-way function [27, 50, 65]) *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is called one way if it satisfies the following*

1. *There is a deterministic polynomial-time Turing machine that on input* α *returns* $f(\alpha)$
2. *For any probabilistic polynomial-time Turing machine* M*, any positive polynomial* $p(n)$ *and sufficiently large* n

$$\Pr(M(f(\alpha), 1^n) \in f^{-1}(\alpha)) < \frac{1}{p(n)},$$

where the probability is taken over all possible choices of $\alpha \in \{0,1\}^n$ *and the internal tosses of* M *conform to a uniform probability distribution. The role of* 1^n *is to allow machine* M *to run in a time polynomial over the length of the pre-image it is supposed to find.*

A stronger notion of unpredictability is that of a hard-core predicate. A polynomial-time computable predicate b is called a hard-core of a function f if all algorithms, given $f(\alpha)$, can guess $b(\alpha)$ only with a probable success which is negligibly better than half.

Definition 11 (hard-core predicate, [27, 50, 65]) *Let* $f : \{0,1\}^* \to \{0,1\}^*$ *and* $f : \{0,1\}^* \to \{0,1\}$*. The predicate* b *is said to be hard-core of the function* f*, if*

1. *There is a deterministic polynomial-time Turing machine that on input* α *returns* $b(\alpha)$*;*
2. *There is no probabilistic polynomial-time Turing machine* M *such that for any positive polynomial* $p(n)$ *and sufficiently large* n

$$\Pr(M(f(\alpha), 1^n) = b(\alpha))) < \frac{1}{2} + \frac{1}{p(n)},$$

where the probability is taken over all possible choices of $\alpha \in \{0,1\}^n$ *and the internal tosses of* M *conform to a uniform probability distribution.*

Theorem 3 (existence of a one-way function with a hard-core predicate, [105, 72, 50]) *If there exists a one-way function, then there exists a one-way function with a hard-core predicate.*

3.5.3 Pseudo-Random Generators

The role of pseudo-random (number) generators (PRNG) in cryptography has already been discussed in Section 2.1.4. Roughly speaking, a PRNG is an efficient (deterministic) algorithm that on input a short seed, outputs a (typically much) longer sequence that is computationally indistinguishable from a uniformly chosen string.

Definition 12 (pseudo-random generator, [27, 50]) *Let* $l : \mathbb{N} \to \mathbb{N}$ *satisfy* $l(n) > n$ *for all* $n \in \mathbb{N}$. *A pseudo-random generator, with a stretch function* $l(n)$, *is a deterministic polynomial time algorithm* G *satisfying the following:*

1. *For every* $\alpha \in \{0,1\}^*$ *it holds that* $|G(\alpha)| = l(|\alpha|)$
2. *The probabilistic ensembles* $\Pi = G(\mathrm{Pr}_0^n)$ *and* $\Pi_0^{p(n)}$ *are computationally undistinguishable.*

Theorem 4 (construction of a pseudo-random generator, [105, 50]) *Let* f *be a one-way* $1:1$ *function, and* b *be a hard-core predicate of* f. *Then*

$$G(\alpha) = b(\alpha)b(f(\alpha))\ldots b(f^{l(|\alpha|)-1}(\alpha))$$

is a pseudo-random generator with a stretch function l.

Consequently, a pseudo-random generator can be constructed from any one-way length-preserving function (rather then merely one-way permutations). On the other hand, the existence of a one-way function is a necessary condition to the existence of the pseudo-random generator, that is

Theorem 5 (existence of pseudo-random generators, [60, 56, 50]) *Pseudo-random generators exists if and only if one-way functions exists.*

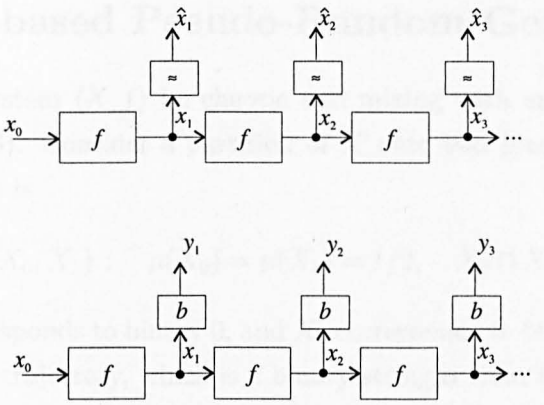


Figure 3.3: A synergy between a chaotic system (top: $\approx$ is a rounding function, $\hat{x_n}$ is the output) and a PRNG (bottom: b is a hard-core predicate, y_n is the output).

Assuming the existence of one-way 1:1 functions, there can exist probability distributions, which are not uniform and are not even statistically close to a uniform distribution, that, nevertheless, are computationally indistinguishable from a uniform distribution [65].

The definition of a pseudo-random generator given above cannot be applied directly since there is no practical way to prove or check rigourously indistinguishability.

Practical cryptography is based on passing known statistical tests (see, for example, [85, 88, 74, 77]), which ensure the pseudo-random property of a generator. Moreover, it is considered that pseudo-random sequences can be used instead of truly random sequences in most cryptographic applications. Chapter 6 describes a basic approach to these statistical tests and evaluates a number of pseudo-chaotic systems.

A parallel between pseudo-random generators and chaotic systems is drawn in Figure 3.3. Though the structure looks similar, there is a fundamental difference: the iterated function of a chaotic system is not required to be one-way. Chaos theory pays no attention to the algorithmic complexity of f and f^{-1}, which is one of the main problems when we apply a chaotic system to cryptography.

3.6 Chaos-based Pseudo-Random Generators

Let the dynamic system $\langle X, f\rangle$ be chaotic and mixing with an f-invariant measure μ (Section 2.2.6). Consider a partition of X onto two μ-equiprobable subsets (Section 3.4.1), that is

$$\beta = \{X_0, X_1\}: \quad \mu(X_0) = \mu(X_1) = 1/2, \quad X_0 \cap X_1 = \emptyset$$

The subset X_0 corresponds to binary 0, and X_1 corresponds to binary 1. The system outputs a symbolic trajectory, which is a binary string α from the initial condition (the seed) $x_0 \in X$. Denote the generator as $G : X \to \{0,1\}^*$. Then

$$G(x) = \alpha = \{s_i\}_{i=1,2,\ldots}, \quad x \in X, \quad s_i \in \{0,1\}.$$

Szczepankski and Kotulski [95, 68] showed that G can produce *asymptotically random* sequences. The next theorem affirms that if we take two different seeds, then with probability one, we obtain two different sequences:

Theorem 6 (the uniqueness of the trajectory, [95]) *For all $x \in X$,*

$$\mu\Big(G^{-1}(\alpha)\Big) = 0$$

.

Ergodically, we expect that the number of zeros in the generated sequence is equal to the expected number of ones. In particular, the Birkhoff-Khinchin Ergodic Theorem [9] can be written as

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \chi_{X_0}\Big(f^i(x)\Big) = \int_{X_0} \chi_{X_0} d\mu = \mu(X_0),$$

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \chi_{X_1}\Big(f^i(x)\Big) = \int_{X_1} \chi_{X_1} d\mu = \mu(X_1),$$

where χ_{X_0}, χ_{X_1} are indicator functions of X_0 and X_1 respectively. Since $\mu(X_0) = \mu(X_1) = 1/2$, the average number of zeros and ones tends to $n/2$.

Theorem 7 (satisfying of the monobit test, [95]) *For all $x \in X$, the number of zeros and ones in α are approximately the same.*

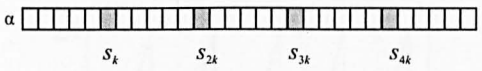


Figure 3.4: A rarefied sample for $k = 6$.

The mixing property ensures the asymptotic independence of bits:

Theorem 8 (asymptotic independence, [95]) *For all $x \in X$ and $n = 1, 2, \ldots$, output bits $s_{(n-1)k}$ and s_{nk} (considered as random variables) are asymptotically independent as k increases, that is*

$$\lim_{k\to\infty} \mu\Big(f^{-nk-k}(X_0) \cap f^{-nk}(X_1)\Big) = \mu\Big(f^{-nk}(X_0)\Big) \cdot \mu\Big(f^{-nk}(X_1)\Big)$$

and

$$\lim_{k\to\infty} \mu\Big(f^{-nk-k}(X_1) \cap f^{-nk}(X_0)\Big) = \mu\Big(f^{-nk}(X_1)\Big) \cdot \mu\Big(f^{-nk}(X_0)\Big).$$

Hence, as k increases, the autocorrelation function of the string $s_k, s_{2k}, \ldots, s_{nk}, \ldots$ (Figure 3.4) tends to a delta function. Practically, the function composition f^k increases nonlinearity of the relationship between x_{nk} and $x_{(n-1)k}$.

So far, by dropping intermediate states from a cryptographic sequence, we improve its pseudo-random appearance. Also, asymptotical independence forms the basis of multi-round block encryption schemes and ensures that the ciphertext is statistically independent of the plaintext.

An Example of Chaos based PRNG

Consider the sawteeth map (Figure 3.5)

$$x_{n+1} = rx_n \bmod q, \tag{3.8}$$

where $x_0 \in [0, q]$, $r = p/q > 1$, and p is co-prime to q. The map is chaotic for all r and has Lyapunov exponent $\lambda = h_{KS} = \log r > 0$.

A discrete version of the sawteeth map is well known in conventional cryptography. The classical Linear Congruental Generator (LCG) is given by

$$x_{n+1} = (Ax_n + C) \bmod M,$$

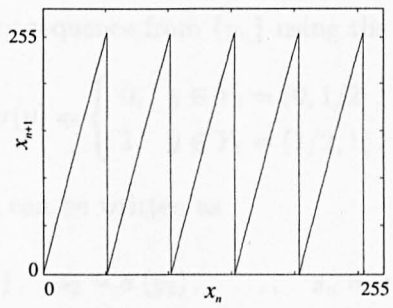


Figure 3.5: The sawteeth map for $p = 1279$ and $q = 255$.

where $x_n \in \{0, 1, \ldots, M\}$ and A, C, M are fixed by the designer [64]. The continuous sawteeth system (and LCG) shows the stretch and folder property of chaos: first, the the state is stretched over a large space (e.g, multiplying or raising in power); then folded q times into a smaller state space (using a periodical function such as mod, sin etc).

However, the continuous sawteeth map, unlike the LCG, cannot tell us about its finite-state approximation, especially about the periodical orbits resulting in patterns.

Theoretically, for large p and q, the sawteeth map is a one-way function: given x_n one can only guess among q equally distributed values what was the previous element x_{n-1}. In each iteration much information is lost, and the inverse sawteeth map is highly multi-valued.

By applying a periodical piecewise-linear function g to each element of the chaotic sequence $\{x_n\}$, i.e.

$$y_0 = g(x_0), \quad y_1 = g(x_1), \quad \ldots, \quad y_n = g(x_n), \quad \ldots$$

we obtain a new chaotic sequence $\{y_n\}$ that is unpredictable in both ways: given y_n, one can only guess among equally distributed values what was the previous element y_{n-1} and what will be the next element y_{n+1}. Periodicity provides a strong fold effect (g^{-1} becomes highly multi-valued), while piecewise-linearity ensures that the uniform distribution of $\{x_n\}$ is passed to $\{y_n\}$. For example, the function g can be another (or the same) sawteeth map.

We can obtain a binary sequence from $\{y_n\}$ using the 1/2 threshold:

$$\sigma(y) = \begin{cases} 0, & y \in Y_0 = (0, 1/2], \\ 1, & y \in Y_1 = (1/2, 1). \end{cases}$$

Finally, the output string can be written as

$$s_1 = \sigma(y_1), \quad s_2 = \sigma(y_2), \quad \ldots, \quad s_n = \sigma(y_n) \quad \ldots$$

Thus, by defining a chaotic function f with control parameters, an additional transformation g and a partition function σ, a pseudo-random generator can be constructed. The combined effect of g and σ is that of a hard-core predicate.

However, the one-step unpredictability of $\{y_n\}$ does not guarantee that the sequence will be unpredictable when an adversary has access to a sufficiently long sequence $\{s_n\}$. In other words, the vast number of samples can theoretically lead to the predictability of $\{s_n\}$.

Chapter 4

Pseudo-Chaotic Cryptographic Systems

This chapter studies cryptographic systems based on finite-state approximations of chaos (i.e. pseudo-chaos). Two approaches are considered to store the system state on a computer: (i) the floating-point format of real numbers and (ii) 'plain' binary strings or m-dimensional cubes.

4.1 Chaos and Pseudo-Chaos

In the previous chapter we have constructed a pseudo-random generator from a chaotic system. This generator produces infinite algorithmically random strings, furthermore, rarefied samples from these strings are asymptotically uncorrelated as the distance k increases. However, this result relates to truly chaotic systems with infinite number of states, whereas digital cryptography is based on finite-state computers.

The notion of *pseudo-chaos* introduced in [34, 26] denotes a computer approximation of chaos. The fundamental differences between mathematical chaos and pseudo-chaos are the following:

- The state variable has a finite length, i.e. stores the state with a finite precision. The system has a finite number of states N.

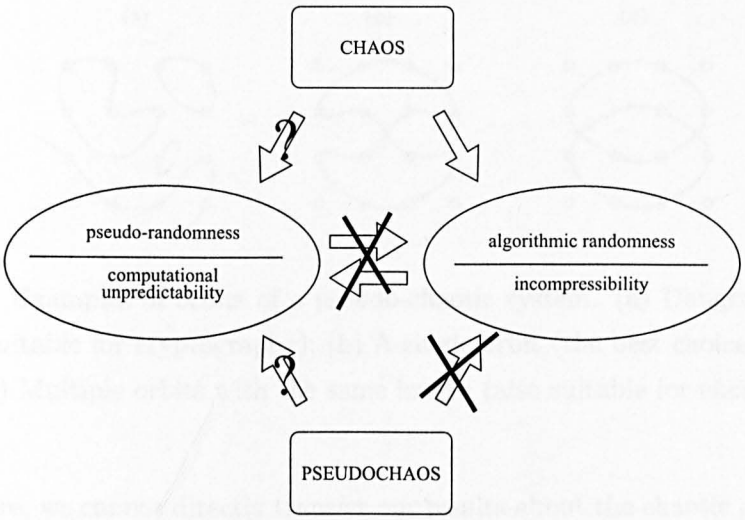


Figure 4.1: Properties of chaotic and pseudo-chaotic systems.

- The iterated function is evaluated with approximate methods, the result is rounded (or truncated) to a finite precision.
- The system is observed during a finite period of time.

The central problem is that rounding is applied in each iteration and the error accumulation causes the original and the approximated processes to diverge very fast. As was first discovered by Lorenz[1], " ... *a small error in the former will produce an enormous error in the future. Prediction becomes impossible* ...". Pseudo-chaos is a poor approximation of chaos since the approximated model does not converge with the original model, and, formally, exhibits non-chaotic properties:

- All trajectories are eventually periodic, i.e. contain patterns. A cycle appears as soon as two states are rounded to the same approximated value.
- Consequently, the Lyapunov exponent λ (Section 2.2.4) and the KS entropy h_{KS} (Section 3.4.2) trivially equal 0.

[1]Lorenz, Edward, is an American meteorologist, who discovered a stable chaotic attractor in the 1960s.

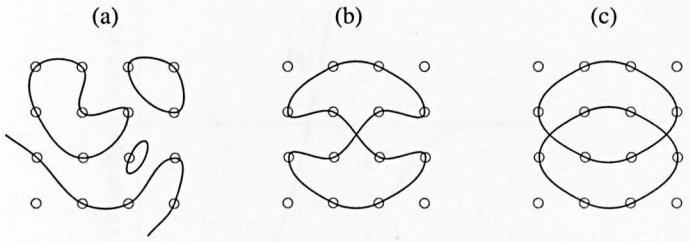


Figure 4.2: Examples of orbits of a pseudo-chaotic system. (a) Dangerously short orbits (unsuitable for cryptography); (b) A single orbit (the best choice for cryptography); (c) Multiple orbits with the same length (also suitable for encryption).

Therefore, we cannot directly transfer our results about the chaotic generator to computer approximations (as summarized in Figure 4.1). The methods from chaos theory should be adopted for pseudo-chaotic systems and cryptographic applications.

What are the minimal, typical and maximal periods of the orbits? Such questions are important in most cryptographic systems. In general, a pseudo-chaotic system produces orbits with different lengths (sometimes called random-length orbits) as illustrated in Figure 4.2-a. Of course, such patterns constitute serious vulnerability: a system may have weak plaintexts and weak keys resulting in recognizable ciphertexts.

If a system has a stable attractor for all initial conditions and parameters, and all orbits have (almost) the same length (Figure 4.2-c), there are more chances to develop a secure encryption scheme. Nevertheless, multiple orbits narrow the search space during cryptanalysis. An ideal cryptosystem has a single orbit passing through the whole state space (Figure 4.2-b).

The next step is to estimated λ of a typical orbit for time not exceeding its period. However, the analysis of periodic orbits depends crucially on the ordering with which the orbits are considered [66]. Two orderings, both corresponding to Lebesgue measure, are considered in the literature: ordering according to the system size N, and ordering according to the minimal period and then lexicographically within the same period.

In the case of the sawteeth map (3.8), with r = 2, two different orderings lead to two opposite answers [66]: ordering by system size yields logarithmic compressibility

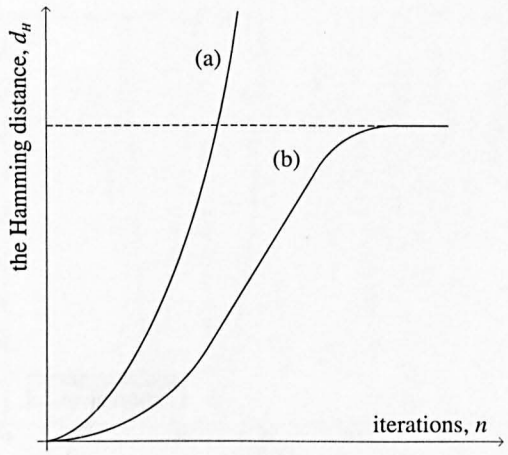


Figure 4.3: A Laypunov exponent of a chaotic (a) and pseudo-chaotic (b) system

of information and zero finite-time (or lack of randomness), while ordering by the minimal period leads to positive finite-time λ and randomness.

A comparison between the average Lyapunov curve of a chaotic system and an analogous property of a pseudo-chaotic system is given in Figure 4.3. If the pseudo-chaotic system has a finite precision σ, then the exponential divergence given by

$$e^{n\lambda} = \frac{|f^n(x_0+\varepsilon) - f^n(x_0)|}{\varepsilon}, \qquad n \to \infty, \varepsilon \to 0, \tag{4.1}$$

will be eventually limited by $\varepsilon = \sigma$. Usually the fraction (4.1) grows exponentially during the first few iterations and then increases linearly until it finally levels off at a certain finite value.

4.2 Floating-point Approximations of Chaos

A floating-point and fixed point arithmetic [58] are the most straightforward solution of approximating a continuous system on a finite state machine (computer). Both approaches imply that the state of a continuous system is stored in a program variable under a finite resolution. A state variable x can be written as a binary fraction $b_m b_{m-1} \ldots b_1 \,.\, a_1 a_2 \ldots a_s$, where a_i, b_j are bits, $b_m b_{m-1} \ldots b_1$ denotes the integer part and $a_1 a_2 \ldots a_s$ is the fractional part of x. Under a finite resolution,

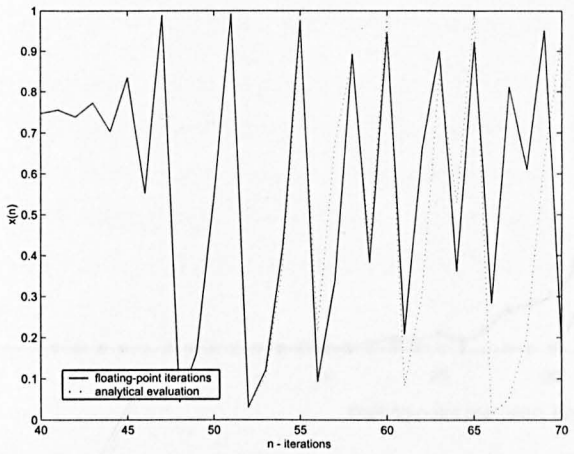


Figure 4.4: Trajectories of a continuous-state chaotic system (4.4) and its 64-bit floating-point approximation. The first curve is obtained by means of the analytical solution (4.5). The rounding off error is amplified in each iteration and the trajectories diverge exponentially.

instead of $x_{n+1} = f(x)$, we write

$$x_{n+1} = round_k(f(x_n)),$$

where $k \leq s$ and $round_k(x)$ is a rounding function defined as

$$round_k(x) = b_m b_{m-1} \ldots b_1 . a_1 a_2 \ldots a_{k-1} (a_k + a_{k+1}).$$

The iterative rounding is accumulative and results in surprisingly different behavior of pseudo-chaos compared with its continuum counterparts. Figure 4.4 shows how fast the original and approximated trajectories diverge.

For cryptographic applications, the rounding off function exposes another danger. Rounding or truncating the state (e.g. to zero values) can lead the process coming out of the chaotic attractor. After that, the system state typically directs to a certain constant value or infinity. Thus, we have to exclude some forbidden initial conditions and parameters, which yield a short orbit or patterns of behavior after a small number of iterations. Figure 4.5 is a plot of the average cycle length verses floating-point precision. It shows that a high precision does not guarantee a sufficiently long trajectory.

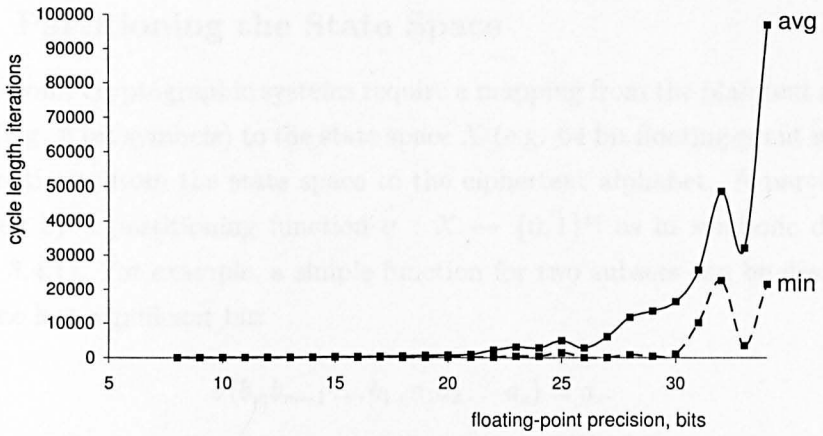


Figure 4.5: The average and the minimal cycle length of the logistic system (4.4) verses floating-point precision obtained from 10 samples of the logistic system.

Another problem is the sensitivity to floating-point processor implementations. Diversified mathematical algorithms or internal precisions in intermediate calculations can lead to the situation when the same code of an encryption software will generate different cryptographic sequences, so there will be no compatibility between software environments.

In Section 3.6 we have mentioned that a chaos-based PRNG with two different seeds produces two different sequence with probability 1. This is true for chaotic systems with an infinite state space, where the probability $\Pr\left(f(x_n) = f(x'_n)\right) \to 0$ with $x_n \neq x'_n$ (despite of the fact that f^{-1} is multi-valued). In finite-state approximations, the probability of mapping two points into one is much higher (for example, in Figure 4.7 $x_n = 0.2$ and $x_n = 0.8$ are mapped to the same point x_{n+1}.). Furthermore, it can occur in each iteration, so that the significant number of trajectories will have identical end routes.

In spite of these shortcomings, a number of investigators have explored the applications of continuous chaos to digital cryptography. In the following Sections, an overview on encryption schemes based on the floating-point approximation of chaos is given.

4.2.1 Partitioning the State Space

Floating-point cryptographic systems require a mapping from the plaintext alphabet $\{0,1\}^m$ (e.g. 8 bit symbols) to the state space X (e.g. 64 bit floating-point numbers) and, sometimes, from the state space to the ciphertext alphabet. A partition can be defined by a partitioning function $\sigma : X \to \{0,1\}^m$ as in symbolic dynamics (Section 3.4.1). For example, a simple function for two subsets can be designed by taking the last significant bit:

$$\sigma(b_m b_{m-1} \ldots b_1 \,.\, a_1 a_2 \ldots a_s) = a_s.$$

If a floating-point system is a pseudo-random generator, the function σ must be irreversible like a hard-core predicate (Section 3.5.2). This can be archived with an equiprobable mapping as suggested in Section 3.6: partitions are selected in such a way, that each symbol occurs with the same probability.

However, it is *not* obligatory to cover all the state space or assign symbols to all partitions. On the contrary, we can change statistical properties of the resulting symbolic trajectory by assigning symbols in a particular way. For example, Figure 4.12 shows a probability distribution of state points in the attractor of a logistic system. Choosing regions with almost the same probability mass, we obtain better statistics in the output.

The number of subsets can be increased, for example, up to 4, 8, 16 etc. In this case the generator will produce more pseudo-random bits per iteration ($m = 2, 3, 4$). Clearly, increasing m reduces the cryptographic strength of the generator since it becomes easier to invert σ.

4.2.2 Chebyshev Map

In 1983 Erber *et al.* [41] suggested using a Chebyshev mixing polynomial to simulate a random process on digital computers. The polynomial is given by

$$x_{n+1} = x_n^2 - 2, \qquad x_n \in (-2, 2)\,. \tag{4.2}$$

The sequence $x_0, x_1, x_2, \ldots$ is chaotic, however, some particular values for x_n must be avoided ($x_n \neq -1, 0, 1$). With a non-integral seed x_0, it is theoretically expected

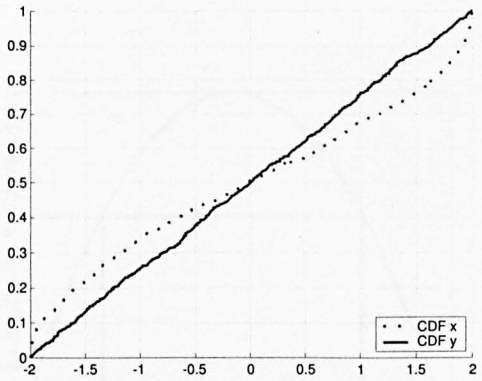


Figure 4.6: The empirical Cumulative Distribution Function of a sequence generated with the Chebyshev system before (doted line) and after (solid line) the additional transformation (4.3). The straight line looks like the uniform distribution on $(-2, 2)$.

that these values will never be encountered. A system with 40-bit precision provides an average cycle of 105 steps [86].

The output y_n is obtained after an additional nonlinear transformation

$$y_n = \frac{4}{\pi} \arccos\left(\frac{x_n}{2}\right) - 2. \tag{4.3}$$

This additional transformation attempts to improve the statistical properties of the sequence, which has a particular probability distribution (the same as for the logistic map given in Figure 4.12).

By applying a transformation with the same shape as the Cumulative Distribution Function (CDF) $\Pr(x_n < x)$ of the input sequence, one can obtain a uniform-like CDF of the output (Figure 4.6). However, the transformation does not affect the *order* of elements, so the Chebyshev sequence is highly correlated.

So far, several investigators (Hosack [59], for example) have found that the Chebyshev transformation possesses undesirable qualities which make it unsuitable for pseudo-random number generation.

4.2.3 Logistic Map

A similar transformation has become of the most famous chaotic maps. In 1976, Mitchell Feigenbaum studied the complex behavior of the so-called logistic map

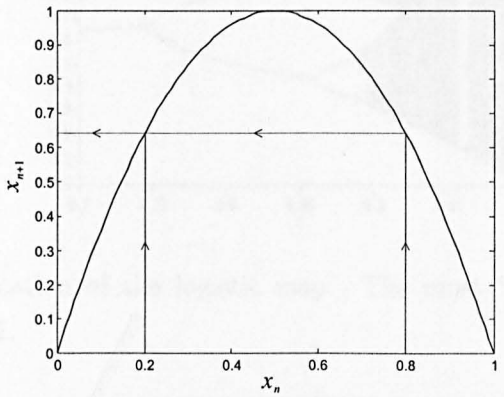


Figure 4.7: The logistic map for $r = 0.99$.

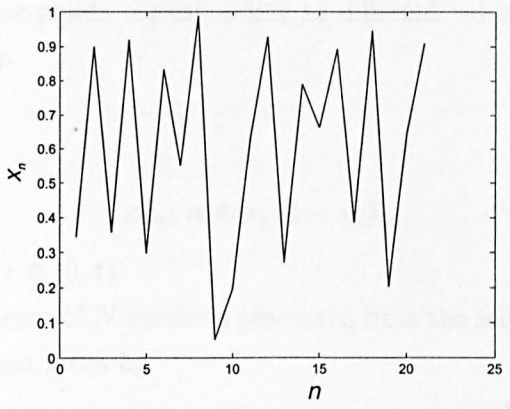


Figure 4.8: A chaotic sequence generated with the logistic map for $x_0 = 0.34$ and $r = 0.99$.

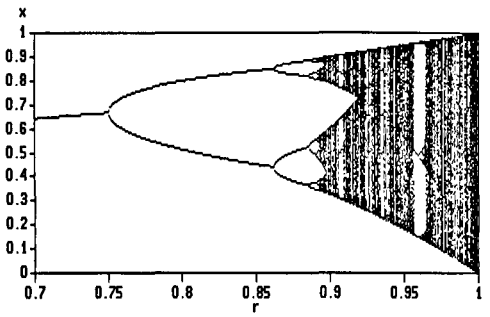


Figure 4.9: Bifurcation of the logistic map. The most 'unpredictable' behavior occurs when $r \longrightarrow 1$.

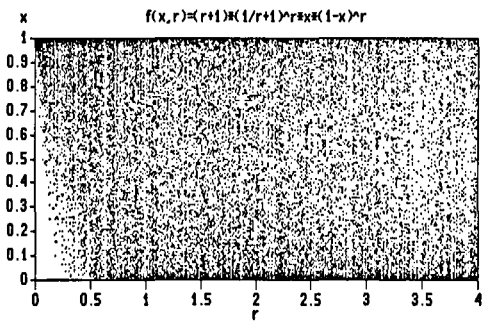


Figure 4.10: Attractor points corresponding to different values of the parameter r in the Matthews map.

(Figure 4.7)

$$x_{n+1} = 4rx_n\left(1 - x_n\right), \tag{4.4}$$

where $x \in (0,1)$ and $r \in (0,1)$.

For any long sequence of N numbers generated from the seed x_0 we can calculate the Lyapunov exponent given by

$$\lambda\left(x_0\right) = \frac{1}{N}\sum_{n=1}^{N} \log\left|r\left(1 - 2x_n\right)\right|.$$

For example, the numerical estimation for $r = 0.9$ and $N = 4000$ is $\lambda\left(0.5\right) \approx 0.7095$.

With certain values of the parameter r, the generator delivers a sequence, which *appears* pseudo-random (Figure 4.8). The Freigenbaum cascade (Figure 4.9) shows

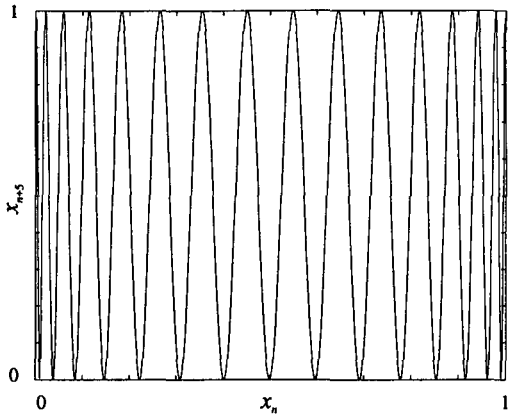


Figure 4.11: The analytical solution at $n = 5$ of the logistic map for $r = 1$.

the values of x_n on the attractor for each value of the parameter r. As r increases, the number of points in the attractor increases from 1 to 2, 4, 8 and infinity. In this area ($r \longrightarrow 1$) it was considered difficult to estimate the final state of the system (without performing n iterations) given initial conditions x_0, or vice-versa - to recover x_0 (which can be a key or a plaintext) from x_n. This complexity was regarded as a fundamental advantage in using continuous chaos for cryptography.

For the boundary value of the control parameter $r = 1$ the analytical solution [62, 51] is:

$$x_n = \sin^2\left(2^n \arcsin\sqrt{x_0}\right). \tag{4.5}$$

When $n = 1$ we have the initial equation (4.4).

Hence, the state x_n can be computed directly from x_0 (without performing n iterations). Figure 4.11 gives the solution for $x_5 = f^5(x_0)$. The number of spikes increases with n, illustrating the sensitivity to the initial conditions. A discussion on the analytical solution is presented in Section 4.2.8.

Bianco *et al.* [25] used the logistic map (4.4) to generate a sequence of floating point numbers, which is then converted into a binary sequence. The binary sequence is XOR-ed with the plaintext, as in the one-time pad cipher (see Section 2.1.3). The parameter r together with the initial condition x_0 form a secret key. The conversion from floating point numbers to binary values is done by choosing two disjoint interval ranges representing 0 and 1. The ranges are selected in such a way,

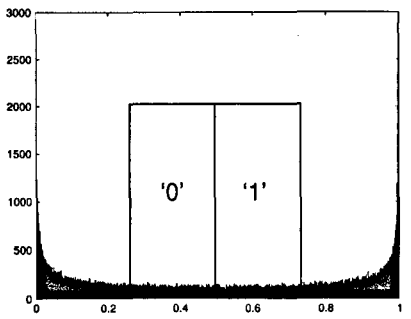


Figure 4.12: The Probability Density Function of a state sequence produced by the logistic system with an incomplete partition.

that the probabilities of occurrence of 0 and 1 are equal (as illustrated in Figure 4.12). Note, the equiprobable mapping does not ensure the uniform distribution. Though the numbers of zeros and ones are equal, the order is not random.

It has been pointed out by Wheeler [99] and Jackson [61] that computer implementations of chaotic systems yield surprisingly different behavior, i.e. it produces very short cycles and trivial patterns (a numeric example in this thesis is given in Figure 4.5).

Matthews [76] generalizes the logistic map with cryptographic constraints and develops a new map to generate a sequence of pseudo-random numbers

$$x_{n+1} = (r+1)\left(\frac{1}{r}+1\right)^{r} \cdot x_n (1-x_n)^r, r \in (1,4).$$

The Matthews system exhibits chaotic behavior for parameter values within an extended range (Figure 4.10) extending the key space. However, no robust cryptographic system was created due to the general floating-point problems, mentioned above.

4.2.4 Tent Map

The one-dimension tent map is defined as

$$x_{n+1} = \begin{cases} a \cdot x_n, & 0 \leq x_n \leq a \\ \frac{1-x_n}{1-a}, & a < x_n \leq 1 \end{cases}, \tag{4.6}$$

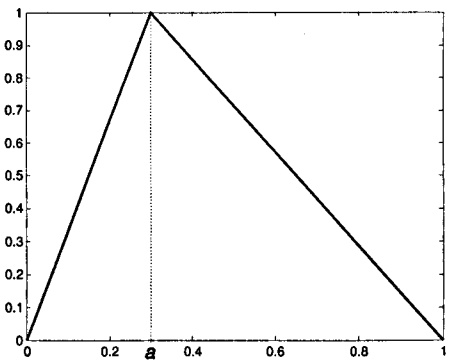


Figure 4.13: The tent map.

where the parameter a determines the top of the tent (Figure 4.13).

The Lyapunov exponent depends on the parameter a and is given by $\lambda(a) = -a\ln(a) - (1-a)\ln(1-a)$ for almost all $x_0 \in (0,1)$ [71].
Numerically, $\max_{a\in(0,1)} \lambda(a) \approx 0.693$ at $a = 0.5$.

The tent map has an analytical solution (again, only for $a = 0.5$) given by

$$x_n = \frac{1}{\pi}\arccos\left(\cos 2^n \pi x_0\right). \tag{4.7}$$

Habutsu *et al.* [53] suggested a chaos-based block encryption system (Figure 2.3) in which the *inverse* of the one-dimensional tent map (4.6) is applied $N = 75$ times to an initial condition representing the plaintext. The 64-bit plaintext is encrypted into 147-bit ciphertext (Figure 2.3). A combination of a floating-point number $a \in [0.4, 0.6]$ and a binary sequence $r = \{r_n\}_{n\in[1,N]}$ is the secret key. The map can be written as

$$x_{n+1} = \begin{cases} a \cdot x_n, & r_n = 0 \\ (a-1)\,x_n + 1, & r_n = 1 \end{cases}, \qquad n \in [1, N]. \tag{4.8}$$

where $r_i \in \{r_i\}_1^N$ and $a \in [0.4, 0.6]$. For N inverse iterations there are 2^N possible ciphertexts which encode the same plaintext.

Biham [24] pointed out that the cryptosystem could be easily broken using a chosen ciphertext type of attack; the complexity of a known plain-text type of attack is 2^{38}.

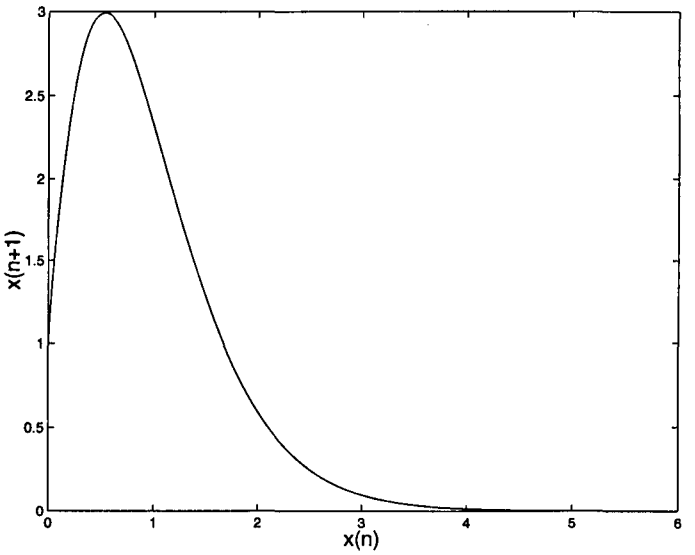


Figure 4.14: The Gallagher map.

4.2.5 Other Chaotic Maps

Gallagher *et el.* [49] developed a chaotic stream cipher, based on the transformation (Figure 4.14)

$$f(x) = \left(a + \frac{1}{x}\right)^{\frac{x}{a}} \qquad x \in (0, 10), \quad a \in [0.29, 0.40].$$

Both the initial condition x_0 and the parameter a represent the key. After $n_0 = 200$ iterations, the system encrypts the plaintext byte p_1 into the ciphertext float $c_1 = f^{n_0+n_1}(x_0)$, i.e. the chaotic map is applied $p_1 \in [0, 255]$ times. Subsequent plaintexts are encrypted using the same trajectory (Figure 2.4-c):

$$c_i = f^{\left(\sum_{i=0}^{p_i} n_i\right)}(x_0).$$

Clearly, disadvantages of such an encryption scheme are: (i) the data expansion (the floating-point representation of c_i is considerably larger that the source byte p_i) and (ii) unstable cycles, incident to floating-point chaos generators.

Kotulski [69] proposes a two dimensional map matching the reflection law of a geometric square and defines conditions under which the system is chaotic and mixing.

There are, in principle, an unlimited number of iterated functions available to generate cryptographic sequences and the nonlinear transformation can be more complex, for example,

$$f(x) = rx\left(1 - \tan\left(\frac{1}{2}x\right)\right), \tag{4.9}$$

$$f(x) = rx\left(1 - \log\left(1 + x\right)\right). \tag{4.10}$$

Though each system has a particular state distribution in the phase space, qualitatively, its behavior is similar to a basic chaotic system such a logistic map.

To increase unpredictability (i.e. the number of states, nonlinearity, complexity) high-order multi-dimensional chaotic system can be used. For example, Paar [80] suggests a second order differential equation describing a robot model

$$m\frac{d^2x}{dt^2} - \beta_2\left(\frac{dx}{dt}\right)^2 \text{sign}\left(\frac{dx}{dt}\right) - \gamma_2 \text{sign}\left(\frac{dx}{dt}\right) - \delta_{21}x - \delta_{23}x^2 = \\ = L\frac{\omega_0^2}{2\pi}\cos\left(\omega_0 t\right) - \zeta_{21}e^{\lambda_{21}t} - \zeta_{22}e^{\lambda_{22}t},$$

which corresponds to a hard spring with coefficients δ_{21} and δ_{23} with two types of friction. The right side of the equation represents a periodic time-dependent force with the amplitude $L\left(\omega_0^2/2\pi\right)$ and a feedback force given by the corresponding parameters.

To date, no such systems have been implemented as a working encryption algorithm (at least no algorithm is known to the author). This is principally due to the relatively complex numerical integration schemes that are required and the non-uniform distribution of state variables.

4.2.6 Multi-Stream Generators

Protopopescu [82] proposes an encryption scheme based on multiple iterated functions: m different chaotic maps are initialized using a secret key. If the maps depend on parameters, these too are determined by the key. The maps are iterated using floating point arithmetic and m bytes are extracted from their floating point representations, one byte from each map. These m numbers are then combined using an

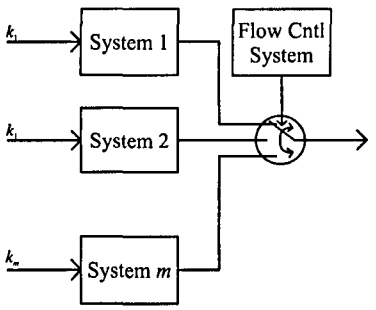


Figure 4.15: A multi-stream cryptographic generator

XOR operation. The process is repeated to create a one time pad which is finally XOR-ed with the plain-text.

In this thesis, we extend the Protopopescu scheme as follows:

1. Chaotic systems can be connected to each other (i.e. the state of each system influences the states of all other systems) to increase the average orbit length. Linked systems form a single chaotic system with a large state space and more stable orbits. A connection between chaotic systems may be established at a given time interval.

2. The set of chaotic systems (iterated functions) can be different for each encryption session. This can be implemented by supplying an iterated function set with the key.

3. The output bit can be generated in each q-th iteration. This increases the independence of bits as suggested in Section 3.6.

4. Chaotic systems can be permuted in a complex manner, particulary, the order of 'switching on' the systems may depend on the key (Figure 4.15).

We can define this extended cryptographic system as

$$\begin{cases} x^1_{n+1} = f_1(x^2_n, k^1), & b^1_j = \sigma_1(x^1_{qj}) \\ x^2_{n+1} = f_2(x^2_n, k^2), & b^2_j = \sigma_2(x^2_{qj}) \\ \cdots & \cdots \\ x^m_{n+1} = f_m(x^m_n, k^m), & b^m_j = \sigma_m(x^m_{qj}) \end{cases}$$

$$b_j = b_j^1 \oplus b_j^2 \oplus \ldots \oplus b_j^m,$$

where

$f_1, f_2, \ldots, f_m$ are iterated functions of the session set
$\langle x_0^1, k^1, x_0^2, k^2, \ldots, x_0^m, k^m \rangle$ are initial conditions; $b_j^1, b_j^2, \ldots b_j^m$ are the internal state bits in $(n = qj)$-th moment of time; b_j is the generator output.

A mixing component providing the property (1) is given by

$$\begin{cases} x_n^1 = mix_1(x_n^1, x_n^2, \ldots, x_n^m) \\ x_n^2 = mix_2(x_n^1, x_n^2, \ldots, x_n^m) \\ \ldots \\ x_n^m = mix_m(x_n^1, x_n^2, \ldots, x_n^m) \end{cases}$$

A pseudo-random generator based on multiple chaotic systems with extended properties (1) – (3) (see page 80) has been implemented in this thesis and forms the basis of the software system E-Larm (see Chapter 5).

For $m = 4$ and $q > 15$ the output sequence is statistically close to a random sequence. Nevertheless, compared with traditional generators, this solution is rather cumbersome and inefficient. It attempts to solve the problems related to the floating-point arithmetic in a 'extensive' way, providing $(m - 1)$ redundant systems (in case one or more systems generate short orbits).

4.2.7 Iteration Counting Ciphers

Baptista [22] and Wong [103] propose a few encryption methods in which the ciphertext is the *number of iterations*. The state space X of a chaotic system is partitioned into m disjoint regions $\{X_1, X_2, ..., X_m\}$ (covering the entire X or, possibly, part of it). A unique plaintext symbol p is assigned to each region. Seeded with an initial condition (x_0, control parameters), the system performs n_0 iterations. All the plaintexts are then encrypted into a sequence of integers

$$c_i = n_i - n_{i-1}, \quad i = 1, 2, \ldots,$$

where n_i is the number of iterations such as the plaintext $p_i = \sigma(X_i)$ and $f^{n_i} \in X_i$. Figure 2.4-b shows the concept of iteration counting ciphers. This encryption scheme

can be extended with a pseudo-random n_{min}, generated for each plaintext symbol. The integer n_{min} defines the minimal number of iteration the system should perform before encrypting the next symbol. Clearly, the disadvantages are: (i) a probability of overflowing the counter $c_i = n_i - n_{i-1}$ (i.e. the length of a trajectory segment can be larger then the maximal number fitting the fixed-length ciphertext c_i); (ii) the ciphertext being larger than the plaintext; (iii) lots of redundant cycles when a long orbit occurs.

Ho [57] continued the research of Baptista and Wong, and went on to develop a software implementation of the encryption techniques, performance evaluation and cryptanalysis. Ho suggests using 4 plaintext bit blocks to speed up the cryptographic algorithm. However, this increases ciphertext redundancy — 4 bits of plaintext are encrypted into a 10 bit ciphertext block. Ho discusses various chaos-specific cryptography issues — dynamic mapping from plaintext to floating-point state, trajectory distribution and algorithm complexity.

4.2.8 Multi-valued Chaotic Maps with Analytical Solutions

A solvable system has an analytical solution of each trajectory, i.e. for any discrete moment of time n we can calculate the state x_n from the initial condition x_0 (without performing n iterations). We have already considered analytical solutions for the logistic (Section 4.2.3) and the tent (Section 4.2.4) systems.

Clearly, an analytical solution of a dynamic system is important in simulation sciences since it eliminates the rounding accumulation problem and speeds up computations. Figure 4.4 gives two trajectories of the logistic system computed using the iterated function (4.4) and the analytical solution (4.5). A special class of solvable systems is very interesting for cryptographic applications. These are one-step unpredictable solvable systems, described by Gonzalez *et al.* [51].

All known solutions for a chaotic system can be written in the general form

$$x_n = \Psi\left(\theta T \kappa^n\right), \tag{4.11}$$

where $\Psi(t)$ is a periodic function with period T, κ is an integer and θ is a real

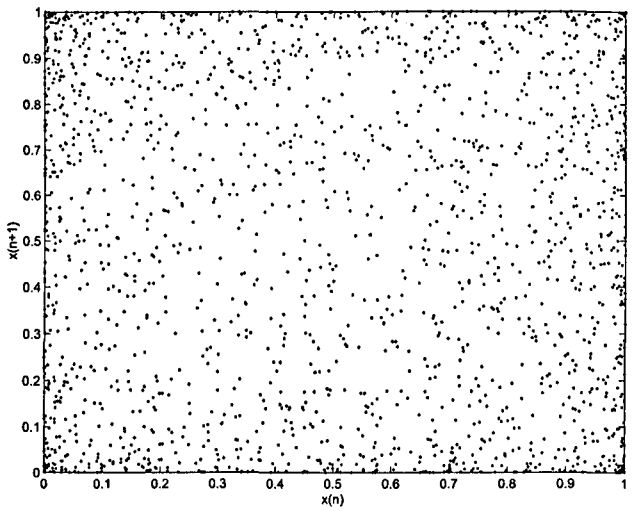


Figure 4.16: A multiple valued chaotic map $x_{n+1} = f(x_n)$

parameter defining the initial condition:

$$x_0 = \Psi(\theta T).$$

The system is chaotic if the Lyapunov exponent $\lambda = \ln \kappa > 0$.

Consider a dynamic system with the solution

$$x_n = \sin^2(\pi\theta\kappa^n). \tag{4.12}$$

The one-step map $x_n \to x_{n+1}$ for this system is given by

$$x_{n+1} = \sin^2(\kappa \arcsin\sqrt{x_n}). \tag{4.13}$$

Unlike the logistic and the tent equations, the map (4.13) can be *multi-valued*, in particular, when κ is an irrational number. Figure 4.16 shows an undefined set of points obtained with the equation 4.12 for $\kappa = \pi^{1/3}$.

On the one hand, analytical solutions provide random access to elements of the cryptographic sequence (unlike conventional pseudo-random generators). On the other hand, the values of x_{n-1} and x_{n+1} cannot be computed from x_n because of the multi-value nature of arcsin in (4.13). So the system is one-step unpredictable (provided the adversary cannot computer the seed and/or does not know the solution).

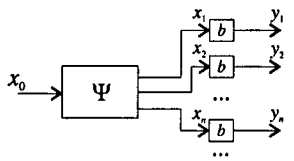


Figure 4.17: A pseudo-random generator based on an analytic solution.

The function $x_n = \Psi(\theta, \kappa, n)$ must have the one-way property to protect the seed x_0. The equation 4.12 has infinitely many solutions for θ and κ given x_n. However, it is difficult to prove that given a long sequence $x_n, x_{n+1}, \ldots$ it is computationally impossible to recover θ (or x_0) and κ. To ensure this, we can introduce a hard-core predicate as illustrated in Figure 4.17.

Unfortunately, analytical solutions have the 'stretch-and-folder' property: first, the arguments (θ, κ) are stretched over a large space (by raising in power and multiplying), and then are folded by means a periodical function (sin, mod). Consequently, the maximum length and the 'pseudo-random' quality of the sequence is limited by the computer precision. As n increases, the effect of an intermediate rounding in (4.12) becomes more destructive, so solvable systems also have a 'horizon' of predictability. For large n, the result (in the context of patterns and probability distributions) may be even worse than iterative computation.

Another problem is that we know analytical solutions only for certain (boundary) values of control parameters. This does not allow to use parameters as a secret key, all the secrecy must be kept in the seed.

Kotulski *et al.* [68] studied floating-point implementations of solvable chaotic systems (4.13) and found them useful for producing short chaotic sequences (e.g. in block ciphers), whereas long sequence did not pass statistical tests. The problem is that generators are very sensitive to the choice of seeds and control parameters, i.e. not all seeds produce a sufficiently long sequence.

4.3 Binary Pseudo-Chaos

Section 2.2.7 provided an introduction to binary chaos, defined by a binary iterated function $f : \Omega \to \Omega$ and a set of *infinite* strings Ω.

Cryptographic systems use finite strings and have a limited exponential divergence of trajectories; in our notation, they are pseudo-chaotic. Typically, binary pseudo-chaotic systems behaves similarly to chaos during the first iterations (e.g. block ciphers, where the number of rounds is, typically, $n = 16$) until the state is truncated. After this, one can discover non-chaotic properties such as periodicity.

A common definition of the Lyapunov exponent for binary systems $\langle \mathcal{A}, f \rangle$, over a finite state space $\mathcal{A}^m \subset \Omega$ is given by

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \log d_H\left(f^n(x_0), f^n(x_0')\right),$$

where $x_0' \in \mathcal{A}$ is taken from the neighborhood of x, i.e. $d_H(x_0, x_0') = 1$.

In a 'good' pseudo-chaotic system of the size 2^m, two trajectories beginning at adjacent points x_0 and x_0' diverge, on average, exactly to $\langle d_H \rangle = m/2$ (Figure 4.3-b).

Like any finite-state system, a binary pseudo-chaotic system is periodical. For example, a Linear Feedback Shift Register (LFSR) given by

$$x_n = (c_1 x_{n-1} + c_2 x_{n-2} + \cdots + c_k x_{n-k}) \bmod 2$$

with particular coefficients c_i [86] has an ideal trajectory of length 2^m.

4.3.1 RANROT

Fog [43] introduced a class of pseudo-chaotic number generators, whose cycle lengths depend on the initial condition. This class of pseudo-chaotic system (named RANROT) is similar to additive generators but with extra rotation or swapping of bits. For example,

$$x_n = ((x_{n-j} + x_{n-k}) \bmod 2^m) \operatorname{rotr} r,$$

where $x_n \in \{0,1\}^m, r \in [0, m/2)$ and j, k are different integers chosen according to certain rules. This nonlinear transformation insures exponential divergence at the beginning of trajectories and unpredictable behavior if the rotation parameter r is unknown. According to the author, the RANROT generators have passed all conventional tests of pseudo-randomness and appear to be faster than other generators of similar quality.

To avoid short cycles, Fog suggests making a periodicity test (up to 1000 iterations) before using a cryptographic application.

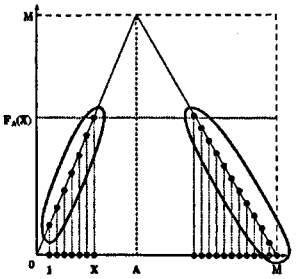


Figure 4.18: The discrete tent map.

4.3.2 Discrete Tent Map

Masuda *et al.* [75] suggest a cryptosystem based on a discretized one-dimension tent map

$$F(X) = \begin{cases} \lceil \frac{M}{A} X \rceil, & 1 \leq X \leq A \\ \lfloor \frac{M}{M-A} (M - X) + 1 \rfloor, & A < X \leq M \end{cases},$$

where $A = 1, 2, ..., M$ and $\lfloor \rfloor, \lceil \rceil$ are round-down and round-up respectively. An initial condition is the plaintext; the final state is the ciphertext and a single parameter is the key. The chaotic map is applied sufficiently many times. The author discusses such properties as the correlation between invariant measures of plaintexts and ciphertexts. Sufficient conditions are given under which the cipher is resistant to differential and linear analysis.

4.3.3 Cellular Automata

Wolfram [102] describes a PRNG based on a one-dimensional cellular automata (CA). The system state is defined by a linear array of elements (cells)

$$\mathbf{b} = (b(1), b(2), \ldots, b(n)) \in \{0,1\}^n.$$

The iterated function $f : \{0,1\}^m \rightarrow \{0,1\}^m$ is given by

$$b(i) = b(i) - 1 \, xor \, \big(b(i) \, or \, b(i+1)\big), \quad \text{for all } 1 \leq i \leq m.$$

The array is considered circular i.e $b(m+1) = b(1)$, and the new element values are considered to be updated in parallel. The output is taken from one cell

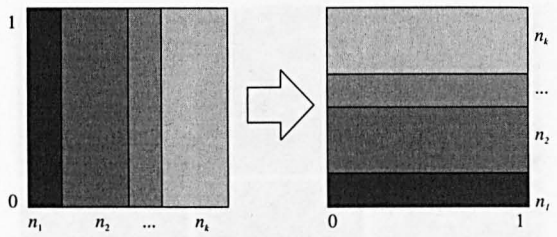


Figure 4.19: A stretch-and-fold baker transformation.

$b_k, k \in [0, n]$ only. This particular CA method requires three array-access operations plus two logical operations for each element of the array. While fairly simple to implement in hardware (since hardware may perform each cell computation in parallel), a large CA system may involve more computations than desired for a software implementation using a single CPU [86].

Gutowitz [52] suggests a complex scheme using a cellular automata. A 512-bit plaintext block is encrypted into a slightly larger ciphertext (578 bits). A two part key (1088-bit) is used for a particular encryption scheme with a so-called block-link structure. There are two rounds, each round consists of two subrounds and each subround includes substitution and permutation phases. The author expects that a sufficient number of rounds ensures a distribution that is statistically close to that of a random process and shows that the cipher is resistant to differential cryptanalysis.

4.3.4 Generalized Baker Map

A generalized Baker map, also known as Kolmogorov flows, represents the most unstable class of chaotic systems, which can be particularly useful for mixing two dimensional data blocks. The iterated function T_ρ can be considered as a geometrical transformation of a square image: the image is divided into vertical strips according to a partition set $\rho = \{n_1, n_2, ..., n_k\}$, stretched horizontally and folded vertically as illustrated in Figure 4.19.

A discrete interpretation of the Baker map based on modular calculus is *invertible*. The iterated function T_ρ is defined on a finite matrix:

$$T_\rho : \{0,1\}^N \rightarrow \{0,1\}^N$$

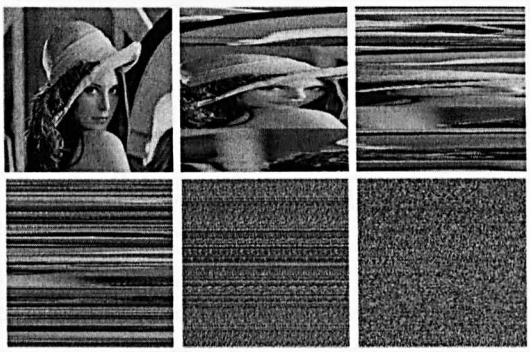


Figure 4.20: Six Baker transformations at $\rho = \{0.25, 0.5, 0.25\}$ applied to Lenna [89].

The set $\{n_1, n_2, ..., n_k\}$ is chosen in such a way that each integer n_i divides N and $n_1 + n_2 + ... + n_k = N$. The bit $b_{r,s} \in \{0,1\}^N$ is mapped to a new position given by

$$T_\rho(r,s) = \left(\frac{N}{n_i}(r - N_i) + s \bmod \frac{N}{n_i}, \frac{n_i}{N}\left(s - s \bmod \frac{N}{n_i}\right) + N_i\right),$$

where $r, s \in \{1, 2, ..., N\}$, $N_i = n_1 + n_2 + ... + n_i$, $i \in \{1, 2, ..., k\}$ and $N_i \leq r < N_i + n_i$. Clearly, this discrete version of the transformation becomes a key-dependent cyclic permutation.

Scharinger [89] and Fridrich [45] describe chaotic block ciphers based on discrete Kolmogorov flows. The technique appears to be highly efficient since large matrices can be permuted in bulk and relatively few iterations scramble the input object to unrecognizable form (Figure 4.20). However, this transformation should be used in combination with a substitution phase, since no permutation can change the statistical properties of the plaintext. With a simple modular addition, Fridrich extends the transformation to three dimensions. The last 3D transformation is a good substitution cipher, which produces ciphertext with a uniform histogram in a few iterations. The diffusion phase is implemented with a LFSR. The resulting encryption scheme is evaluated in terms of resistance to the general classes of attacks, and no cryptographic weakness is found. Cappelletti [32] designed a hardware implementation (using FPGA) of the Scharinger's scheme.

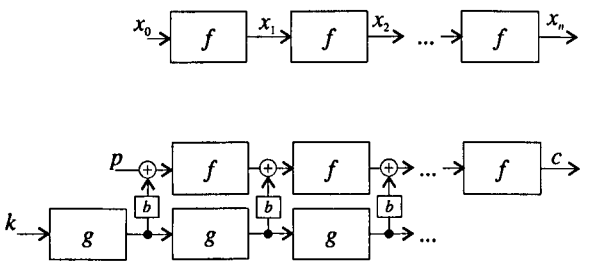


Figure 4.21: A typical block cipher is a combination of several pseudo-chaotic systems

4.3.5 Pseudo-Chaos and Conventional Cryptosystems

Pseudo-Random Generators

Existing pseudo-random generators can be viewed as pseudo-chaotic systems as illustrated in Section 3.6. Another example is the Blum-Blum-Shub system [77, 86] given by the iterated function

$$x_{n+1} = x_n^2 \bmod M,$$

where $M = pq$, p, q are two distinct prime numbers each congruent to 3 modulo 4. The output bit b_n is obtained from a predicate $\sigma(x_n)$, which is the last significant bit of x_n.

Besides the sensitivity to the initial condition and the topological transitivity, a pseudo-random generator has to be computationally unpredictable. The last property is ensured by a *one-way* iterated function and a hard-core predicate (Section 3.5.2). A one-way transformation is based on a certain mathematical problem, which is considered unsolved. For example, the Blum-Blum-Shub function works under the assumption that integer factorization is intractable.

Chaos theory is not focused on the algorithmic complexity of the iterated function, whereas in cryptography the complexity is the key issue, i.e. security.

Symmetric Block Ciphers

All classical iterative block ciphers, in our notation, are pseudo-chaos or combinations of several pseudo-chaotic systems. As an example, consider the new Rijndael algorithm [84] accepted as AES[2]. The system state x is a two-dimensional array of bits. The plaintext is assigned to the initial conditions x_0, and after a fixed number of iterations ($n = 10 \ldots 14$), the ciphertext is obtained from the final state x_n. The encryption transformation is a combination of several pseudo-chaotic maps:

1. the substitution phase is a composition of multiplicative inverse and affine transformations;
2. the mixing phase includes cycle shifts and column multiplication over a finite field;
3. the round key is obtained from another pseudo-chaotic system (pseudo-random generator).

If we consider the substitution and mixing phases as a single iterated function, the encryption scheme will represent two linked pseudo-chaotic systems (Figure 4.21).

[2]The Advanced Encryption Standard is an encryption algorithm for securing sensitive but unclassified material by US Government agencies and, as a likely consequence, may eventually become the de facto encryption standard for commercial transactions in the private sector.

Chapter 5

E-Larm: A Chaos Based Encryption System

5.1 Overview

E-Larm is a cryptographic tool designed to evaluate chaos-based algorithms. It allows the generation of pseudo-chaotic sequences from a floating-point seed as well as encrypting and decrypting binary files. E-Larm provides an implementation of a multi-stream generator as described in Section 4.2.6. The encryption scheme is a sample Vernam cipher (Section 2.1.3) using a pseudo-chaotic sequence as the keystream. The software is given on the CD in Appendix A.

The analysis of the output (pseudo-chaotic sequences, ciphertext) is undertaken in mathematical packages such as Matlab 6 and Statistica 6 as well as statistical test suite developed by NIST [88].

5.2 Algorithm

The basic idea of this generator is to extract bits from multiple pseudo-chaotic systems in a complex and key-dependent manner. E-Larm code contains a set of one-dimensional iterated functions $f : X \rightarrow X$, all producing chaos on a certain domain (typically, $(0, 1)$). The number of iterated function used in the same session is $\mathtt{Ns} = 4$. Examples of the functions are given below (their properties are assessed

in Chapter 6):

1. Smooth nonlinear functions
   - Logistic map $x_{n+1} = px_n(1 - x_n)$, $p = 3.9$
   - Sine map $x_{n+1} = |\sin(p\pi x_n)|$, $p = 5.0$
   - Tangent-feedback map $x_{n+1} = 3.3x_n(1 - \tan(1/2x_n))$
   - Logarithm-feedback map $x_{n+1} = px - n(1 - \log(1 + x_n))$, $p = 5.0$
2. Piecewise-linear functions
   - Tent map $x_{n+1} = 1 - |2x_n - 1|$
   - Sawteeth map $x_{n+1} = 5x \bmod 4$

For each iterated function we define a 2-region partitioning function `part` : $\mathbb{X} \rightarrow \{0, 1\}$ (section 3.4.1) in such a way that the state point spends, on average, equal periods of time in each region. If the state point does not belong to any region, the function `part` returns -1. Practically, `part` can be defined using one of two approaches:

1. **Last Significant Bits (LSB)**
   The function `part` extracts one or more last significant bits from the floating point format of the state variable x. Then the bits can be combined into one output bit, for example, using the XOR-operator. The E-Larm implementation of `part` extracts two last bits (for certain maps the single bit has biased statistics because of the rounding):

$$\texttt{part}(x) = (x \, and \, 1) \, xor \, \big((x \, xor \, 2) \gg 1\big),$$

   where $\gg$ is the right shift. This approach is very efficient, especially when the state space is non-uniform (e.g logistic, sine, logarithm, tangent maps) because the space is partitioned into a large number of regions lying on the resolution grid. If an area of the state space has a higher probability mass, then it is equally distributed among small regions covering this area.

2. **Threshold Partitioning (P2)**

   The second approach is to divide the state space into two large compact regions. The boundary values are selected using the symmetry of the probability density function of the system states (e.g. Figure 4.12). Symbolically, that is

$$\texttt{part}(x) = \begin{cases} 0, & X^0_{min} \leq x \leq X^0_{max} \\ 1, & X^1_{min} \leq x \leq X^1_{max} \\ -1, & \text{otherwise} \end{cases},$$

   where $[X^0_{min}, X^0_{max}] \cup [X^1_{min}, X^1_{max}] \subset (0,1)$ and $[X^0_{min}, X^0_{max}] \cap [X^1_{min}, X^1_{max}] = \emptyset$. If $X^0_{max} = X^1_{min} = X_{thres}$; we then have a simple threshold partitioning, or, in other words, first significant bit partitioning. Practically, this approach is more efficient for piecewise-linear maps, such as the tent or sawteeth maps, because their floating point approximation produces patterns in the last significant bits.

The following table describes variables and functions referenced in the algorithm.

Built-in Variables & Functions		
`Ns`	`int`	number of the pseudo-chaotic systems
`MAXiter`	`int`	minimal and maximal number of intermediate states
`func[Ns]`	`func[]`	set of iterated functions of the form `state[s] = func(state[s], param[s])`
`part[Ns]`	`func[]`	set of partitioning functions of the form `b = func(state[s])`
`mix`	`func`	mixing function of the form `state[s] = mix(s)`
Input Variables		
`L`	`int`	length of the output sequence
`p[Ns]`	`int[]`	parameters of the systems
`seed[Ns]`	`double[]`	initial states
`MINiter`	`int`	minimal number of intermediate states
Output Variables		
`out[L]`	`byte`	the output sequence
Internal Variables		
`s`	`int`	counter of the systems
`i`	`int`	counter of output symbols
`j`	`int`	counter of intermediate states
`state[Ns]`	`double[]`	current states of the systems
`b`	`byte`	bit produced by a pseudo-chaotic system

Algorithm 5.1 describes a basic version of the E-Larm generator in a pseudo-code. The generator produces a binary sequence `out[L]`. A combination of initial states `seed[]`, control parameters `p[]` and `MINiter` may be used as a secret key (generator seed). However, the values should be chosen from an appropriate range for each system.

A minimal number of chaotic iterations `MINiter` per round is defined to control the dependence of bits in the output sequence (see *asymptotic independence* in

Section 3.6). To avoid the short cycles, at least at some degree, we suggest to *mix* periodically the states of pseudo-chaotic systems used. In E-Larm, the function `mix` is applied to the state of the s-th system, setting an influence of all systems on the current system:

$$\texttt{mix(s)} = \left(\texttt{state[s]} \sum_{q=1\ldots N_s, q\neq s} \texttt{state[q]} \right) \bmod 1,$$

i.e. the system state is multiplied by the sum of other system states, and the fractional part of the result is taken.

The upper limit `MAXiter` is necessary to treat 'bad trajectories' that come out of the chaotic attractor due to floating-point rounding. In this case the function `part` becomes -1 for all subsequent iterations and we reset the system with the `mix` function.

Typical values of parameters are given below:

`Ns = 4` `MINiter = 10` `MAXiter = 20`

5.3 Practical Implementation

5.3.1 Platform

The software platform of E-Larm is Java 2 version 1.4. The choice of platform was determined by the following:

- Java ensures portability of cryptographic algorithms using floating-point arithmetic. The class `StrictMath` contains methods for performing basic numeric operations such as the elementary exponential and trigonometric functions independently from the hardware architecture.
- Java is the most advanced developing tool of cryptographic applications. The Java Cryptography Extension (JCE) provides a framework and implementations for encryption, key generation and other algorithms.

5.3.2 User Interface

E-Larm has both a command line and graphical interface:

Algorithm 5.1: Multi-Stream Pseudo-Chaotic Generator

```
// Initialization
for ( s = 0 ; s < Ns ; s = s + 1 )   // for each system
    state[s] = seed[s]               // assign the initial condition to
                                     // the state variable
end for

// Main cycle
for ( i = 0 ; i < L ; i = i + 1 )                // for each output bit
    for ( s = 0 ; s < Ns ; s = s + 1 )           // for each system
        state[s] = mix(s);                       // mix the s-th system states
        for ( j = 0 ; j < MINiter ; j = j + 1 )  // iterate MINiter times
            state[s] = func[s]( state[s], p[s] )  // evaluate the chaotic map
        end for
        do
            state[s] = func[s]( state[s], p[s] )  // evaluate the chaotic map
            if ( j > MAXiter )       // bad trajectory?
                state[s] = 1         // assign 1 to produce a nonzero mix seed
                state[s] = mix(s)    // reset the system with a mix seed
            end if
            b = part( state[s] )     // obtain a bit using a partitioning functi
            j = j + 1                // increase the counter
        while ( b == -1 )            // loop until the state point returns
                                     // to the working domain or
                                     // the counter j exceed MAXiter
        out[i] = (out[i] + b) mod 2 // append XOR'ed bits coming from
                                     // the systems to the output string
    end for
end for
```

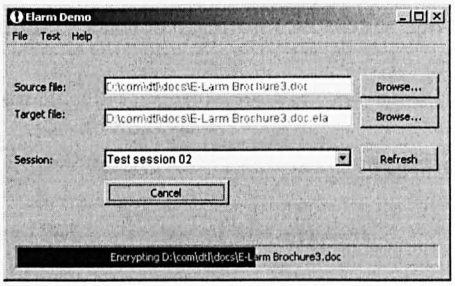


Figure 5.1: E-Larm GUI

- In command line mode, the application accepts up to 3 arguments:
  `java ElarmDemo [InputFileName [OutputFileName [SessionName]]]`

  E-Larm encrypts (decrypts) the file `InputFileName` into `InputFileName` using session `SessionName`. If the extension of `InputFileName` is different from `ela`, the software performs encryption, otherwise (if the input file extension is `ela`) it performs decryption. If the number of arguments is less than 3, the graphical user interface is lunched and initialized with arguments, if given.

- E-Larm has a graphical user interface based on the class library `javax.swing.*` (Figure 5.1). Input and output files can be entered manually or by means of a file browser (`Browse...`). Push button `Encrypt` (`Decrypt`) activates a separate thread performing the cryptographic transformation. The tread can be stopped with the button `Cancel`.

  Additionally, one can save a pseudo-random sequence to a binary file, generated from the current session by choosing `Test` → `Generate Session...`

5.3.3 Sessions

An encryption (decryption) session describes a set of pseudo-chaotic systems used by the multi-stream generator. For each iterated function we define initial conditions (seed), control parameters and algorithm parameters (minimal number of iterations and a partitioning function). Session defines the environment in which the cryptographic sequence is produced and can be considered as a key.

The following table defines session variables and their data types used by E-Larm:

Properties of `session`		
`id`	`int`, $[1, 2^{15}]$	ID of the session
`name`	`char[255]`	name of the session
Properties of `iterfunc`		
`id`	`int`, $[1, 2^{15}]$	ID of the iterated function
`seed`	`double`, $[0, 1]$	the initial condition of the iterated function
`miniter`	`int`, $[1, 10^4]$	minimal number of intermediate iterations time
`lowerbound`, `upperbound`	`double`, $[0, 1]$	lower and upper bounds of a state space subset from which a cryptographic sequence a produced. These parameters define the partitioning function `part`: the interval `[lowerbound, upperbound]` is equally divided in two region corresponding to logical 0 and 1. This parameters are ignored if the LSB partitioning function is used.

For the purpose of the usability, sessions are stored in the XML format as illustrated below. Once the E-Larm application is started, sessions are loaded from `conf/sessions.xml` and becomes available in the `Session` combo box (Figure 5.1). The `Refresh` button causes the sessions to be reloaded from the file.

Source 5.1: An XML session file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Elarm Session Keys -->
<!DOCTYPE sessions SYSTEM "sessions.dtd">

<sessions>

<session id="1" name="Test Session 01">

<iterfunc id="1" seed="0.26" miniter="10"
          lowerbound="0.2" upperbound="0.8"/>

<iterfunc id="2" seed="0.33" miniter="12"
          lowerbound="0.18" upperbound="0.9"/>

<iterfunc id="3" seed="0.745" miniter="8"
          lowerbound="0.6" upperbound="1"/>
</session>

<session id="2" name="Test Session 02">

<iterfunc id="2" seed="0.87" miniter="12"
          lowerbound="0.18" upperbound="0.9"/>

<iterfunc id="3" seed="0.61" miniter="8"
          lowerbound="0.6" upperbound="1"/>

<iterfunc id="4" seed="0.589" miniter="10"
          lowerbound="0.25" upperbound="0.8"/>

</session>

</sessions>
```

5.4 Initial Conditions and Parameters

Chaotic generators are sensitive to the choice of initial conditions and parameters. If the initial condition x_0 does not belong the chaotic (strange) attractor, the system can reach a point attractor or periodic attractor, and start generating a trivial sequence. It is important to ensure that the control parameters correspond a chaotic behaviour.

In E-Larm the range for the initial condition x_0 and the value of parameter p were defined in the following way:

- For well known chaotic map (logistic, sine, tent, sawteeth maps), the values were taken to guarantee most unstable mode as described in the literature (e.g. [81], [20]).
- Original chaotic functions (tangent, logarithm) were assigned parameters by studying these functions analytically and by carrying simulations in Matlab. In particular, histograms (PDF of the state space) were studied at different control parameters (histogram plots are given in Section 6.5).

Following is a list of iterated functions with initial conditions and control parameters used by E-Larm. Initial conditions (seed) are defined in the session file (Section 5.3.3), whereas control parameters are built in the application (i. e. defined in the source code).

Table 5.1: Continuous nonlinear maps

Name	Equation	Params	Init. Cond.
Logistic map	$x_{n+1} = px_n(1 - x_n)$	$p = 3.99$	$x_0 \in (0, 1)$
Sine map	$x_{n+1} = \lvert \sin(p\pi x_n) \rvert$	$p = 5.0$	$x_0 \in (0, 1)$
Tangent-feedback map	$x_{n+1} = px_n(1 - \tan(0.5x_n))$	$p = 3.3$	$x_0 \in (0.5, 1.5)$
Logarithm-feedback map	$x_{n+1} = px_n(1 - \log(1 + x_n))$	$p = 5.0$	$x_0 \in (0.3, 1.5)$
Exactly solvable map	$x_n = \sin^2(\pi\theta\kappa^n)$	$\kappa = \pi^{1/3}$	$x_0 = (0, 1)$

Table 5.2: Piecewise-linear maps

Name	Equation	Params	Init. Cond.
Tent map	$x_{n+1} = 1 - \|px_n - 1\|$	$p = 2$	$x_0 \in (0, 1)$
Sawteeth map	$x_{n+1} = p/qx_n \bmod q$	$p = 5,\ q = 4$	$x_0 \in (0, 1)$

5.5 Evaluation

Pseudo-randomness and performance tests are described in Chapter 6.

Chapter 6

Evaluation of Chaos-based PRNG's used for E-Larm

6.1 Introduction

Various statistical tests [64, 77, 85, 88, 74, 98] can be applied to a given chaotic sequence to compare and evaluate it to a truly random sequence (section 3.3.1). The likely outcome of statistical tests, when applied to a truly random sequence, is known *a priori* and can be described in probabilistic terms. There are an infinite number of possible statistical tests, each assessing the presence or absence of a pattern which, if detected, would indicate that the sequence is non-random, but no specific finite set of tests is deemed 'complete'. In addition, the results of statistical testing must be interpreted with some care and caution to avoid incorrect conclusions about a specific generator.

In this research we borrow the test methodology described in [88]. A classical statistical test is formulated to test a specific null hypothesis H_0. Let the null hypothesis be that the sequence being tested is *random.* Associated with this null hypothesis is the alternative hypothesis H_a, which is that the sequence is *not random.* For each applied test, a decision or conclusion is derived that accepts or rejects the null hypothesis, i.e., whether the generator is (or is not) producing random values,

based on the sequence that was produced.

For each test, a relevant randomness statistic must be chosen and used to determine the acceptance or rejection of the null hypothesis. Under an assumption of randomness, such a statistic has a distribution of possible values. A theoretical reference distribution of this statistic under the null hypothesis is determined by mathematical methods. From this reference distribution, a critical value is determined (typically, this value is 'far out' in the tails of the distribution, say out at the 99% point). During a test, a test statistic value is computed on the data (the sequence being tested). This test statistic value is compared to the *critical value*. If the test statistic value exceeds the critical value, the null hypothesis for randomness is rejected. Otherwise, the null hypothesis (the randomness hypothesis) is *not* rejected (i.e. the hypothesis is accepted).

In practice, the reason that statistical hypothesis testing works is that the reference distribution and the critical value are dependent on and generated under a tentative assumption of randomness. If the randomness assumption is, in fact, true for the data at hand, then the resulting calculated test statistical value on the data will have a very low probability (e.g., 0.01%) of exceeding the critical value.

On the other hand, if the calculated test statistical value *does* exceed the critical value (i.e., if the low probability event does in fact occur), then from a statistical hypothesis testing point of view, the low probability event should not occur naturally. Therefore, when the calculated test statistical value exceeds the critical value, the conclusion is made that the original assumption of randomness is suspect or faulty. In this case, statistical hypothesis testing yields the following conclusions: reject H_0 (randomness) and accept H_a (non-randomness).

Statistical hypothesis testing is a conclusion-generation procedure that has two possible outcomes, either accept H_0 (the data is random) or accept H_a (the data is non-random). The following table relates the true (unknown) status of the data at hand to the conclusion arrived at using the testing procedure:

True situation	Conclusion	
	Accept H_0	**Accept H_a**
Data is random (H_0 is true)	No error	Type I error
Data is not random (H_a is true)	Type II error	No error

If the data is, in truth, random, then a conclusion to reject the null hypothesis (i.e., conclude that the data is non-random) will occur a small percentage of the time. This conclusion is called a Type I error. If the data is, in truth, non-random, then a conclusion to accept the null hypothesis (i.e., conclude that the data is actually random) is called a Type II error. The conclusions to accept H_0 when the data is really random, and to reject H_0 when the data is non-random, are correct.

The probability of a Type I error is often called the *level of significance* of the test. This probability can be set prior to a test and is denoted as α (`ALPHA`). For the test, α is the probability that the test will indicate that the sequence is not random when it really is random. That is, a sequence appears to have non-random properties even when a 'good' generator produced the sequence. Common values of α in cryptography are about 0.01.

The probability of a Type II error is denoted as β. For the test, β is the probability that the test will indicate that the sequence is random when it is not; that is, a 'bad' generator produces a sequence that appears to have random properties. Unlike α, β is not a fixed value. β can take on many different values because there are an infinite number of ways that a data stream can be non-random, and each different way yields a different β. The calculation of the Type II error β is more difficult than the calculation of a because of the many possible types of non-randomness.

One of the primary goals of the following tests is to minimize the probability of a Type II error, i.e., to minimize the probability of accepting a sequence being produced by a good generator when the generator was actually bad. The probabilities α and β are related to each other and to the size N of the tested sequence in such a way that if two of them are specified, the third value is automatically determined. In practice, we usually select a sample size N and a value for α. Then a critical point for a given statistic is selected that will produce the smallest β. That is, a suitable sample size is selected along with an acceptable probability of deciding that

a bad generator has produced the sequence when it really is random. Then the cutoff point for acceptability is chosen such that the probability of falsely accepting a sequence as random has the smallest possible value.

Each test is based on a calculated test statistic value, which is a function of the data. If the test statistic value is S and the critical value is t, then the Type I error probability is $P(S > t|H_0 \text{ is true}) = P(\text{reject } H_0|H_0 \text{ is true})$, and the Type II error probability is $P(S \leq t|H_0 \text{ is false}) = P(\text{accept } H_0|H_0 \text{ is false})$. The test statistic is used to calculate a *P-value* that summarizes the strength of the evidence against the null hypothesis. For these tests, each *P-value* is the probability that a perfect random number generator would have produced a sequence 'less random' than the sequence that was tested, given the kind of non-randomness assessed by the test.

If a *P-value* for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A *P-value* of zero indicates that the sequence appears to be completely non-random. A significance level α can be chosen for the tests. If *P-value* $\geq \alpha$, then the null hypothesis is accepted; i.e., the sequence appears to be random. If *P-value* $< \alpha$, then the null hypothesis is rejected; i.e., the sequence appears to be non-random. The parameter α denotes the probability of the Type I error. Typically, α is chosen in the range $[0.001, 0.01]$.

Most of the tests have the standard normal and the chi-square χ^2 as reference distributions. If the sequence under test is in fact non-random, the calculated test statistic will fall in extreme regions of the reference distribution. The standard normal or Gaussian distribution

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

is used to compare the value of the test statistic obtained from the RNG with the expected value of the statistic under the assumption of randomness. The test statistic for the standard normal distribution is of the form $z = (x - \mu)/\sigma$, where x is the sample test statistic value, and μ and σ^2 are the expected value and the variance of the test statistic. The χ^2 distribution is used to compare the goodness-of-fit of the observed frequencies of a sample measure to the corresponding expected frequencies of the hypothesized distribution. The test statistic is of the form $\chi^2 = \sum \left((o_i - e_i)^2 / e_i\right)$, where o_i and e_i are the observed and expected frequencies of occurrence of the measure, respectively.

6.2 Considerations for Randomness and Unpredictability

The following assumptions are made with respect to random binary sequences to be tested:

Uniformity: At any point in the generation of a sequence of random or pseudorandom bits, the occurrence of a zero or one is equally likely, i.e., the probability of each is exactly 1/2. The expected number of zeros (or ones) is $N/2$.

Scalability: Any test applicable to a sequence can also be applied to subsequences extracted at random. If a sequence is random, then any such extracted subsequence should also be random. Hence, any extracted subsequence should pass any test for randomness.

Consistency: The behavior of a generator must be consistent across initial conditions (seeds). It is inadequate to test a PRNG based on the output from a single seed, or a generator on the basis of an output produced from a single physical output.

6.3 Chaotic Maps

The following tables describe chaotic maps that have been tested. The finite state approximation was the 64-bit floating-point format (`double`). Two different partitioning methods were used: last significant bit (LSB) for continuous nonlinear maps, and threshold partitioning (2 regions) for piecewise-linear maps (see sections 5.2 and 3.4.1). This choice was determined experimentally as will be discussed later.

Table 6.1: Continuous nonlinear maps

Name	Code	Equation	Params	Fig.
Logistic map	par	$x_{n+1} = px_n(1 - x_n)$	$p = 3.99$	6.1
Sine map	sin	$x_{n+1} = \lvert \sin(p\pi x_n) \rvert$	$p = 5.0$	6.3
Tangent-feedback map	tan	$x_{n+1} = px_n(1 - \tan(0.5x_n))$	$p = 3.3$	6.4
Logarithm-feedback map	log	$x_{n+1} = px_n(1 - \log(1 + x_n))$	$p = 5.0$	6.5
Exactly solvable map	exactsol	$x_n = \sin^2(\pi\theta\kappa^n)$	$\kappa = \pi^{1/3}$	6.9

Table 6.2: Piecewise-linear maps

Name	Code	Equation	Params	Fig.
Tent map	tent	$x_{n+1} = 1 - \lvert px_n - 1 \rvert$	$p = 2$	6.6
Sawteeth map	saw	$x_{n+1} = p/qx_n \bmod q$	$p = 5$, $q = 4$	6.7

The E-Larm mixing scheme (referenced hereafter as 'combo' system) has been tested on 4 pseudo-chaotic systems: logistic, sine, tangent and sawteeth maps.

6.4 Visual Approaches

Before starting numeric tests it is helpful to check the PRNG behavior using simple plots in different domains: space-time, phase-space and frequency as well as to check the time series distribution.

6.4.1 Phase-Space Diagrams

Traditionally, a chaotic system is defined by an iterated function of the form $x_{n+1} = f(x_n)$ as introduced in section 2.2. Most of chaotic maps, that have been studied, are simple and easily predictable (first row in Figures 6.1, 6.3, 6.4, 6.5, 6.6, 6.7, 6.9). However, the 15-round composition $x_{n+15} = f^{15}(x_n)$ shows the complexity and mixing effect of chaos. Such a multi-round transformation ensures the asymptotic independence of the output bits (section 3.6), i.e hides the time series correlations and increases the computational unpredictability.

6.4.2 Time series

The second row in Figures 6.1, 6.3 etc depicts a time series and its 1000-bin histogram (approximated probability density function). Visibly, the histograms of the floating-point numbers are not flat, i.e. the system states are not uniformly distributed over the state space and, consequently, are predictable. However, histograms of extracted bits using a *convenient* partition method appear to be uniform:

- In the case of a piecewise linear map (sawteeth, tent maps) the statistical result is better when we choose two or more compact non-overlapping regions (e.g. using a threshold). Piecewise linear maps can produce time series with good randomness in higher order bits, but often keep the last bit(s) unmodified (as illustrated in Figure 6.8), even with control parameters that should theoretically guarantee chaos.

- On the other hand, for a continuous nonlinear map (logistic, sine maps), the last significant bit (LSB) method provides more uniform statistics because of a rounding effect (Figure 6.2). The LSB method works even when the histogram is not symmetric and it is difficult to find two compact partitions where total probability masses are equal (e.g. Figures 6.4 and 6.5).

The time series obtained with an analytical (exact) solution (section 4.2.8) have different histograms in the beginning and at the end of the sequence (Figure 6.9) because the precision degrades. The last approximated PDF is not continuous, which illustrates the rounding and approximating artifacts. The cycle length histogram is given in Figure 6.10.

6.4.3 Power Spectrum

The spectral density Fourier transform function detects periodic features. The spectral density function (spectrum) is given by

$$P_m = \left(\frac{1}{N} \sum_{n=-N/2}^{N/2} x_n \exp(-i2\pi nm/N) \right)^2 ,$$

where N is the length of the sequence, m is the spatial frequency.

It is considered that the spectrum should be uniform and no more than 5% of the peaks should surpass the 95% cutoff. Clearly, most of chaotic maps have a non-unform spectrum of their time series. But, if we drop out $k = 15$ intermediate states (multi-round composition), the new sequence will have a 'more random' spectrum (bottom row of Figures 6.1, 6.3 etc).

6.4.4 Approximated Entropy

Approximated entropy (APEN) characteristics are based on repeating patterns in the string. "For a fixed block length m, APEN(m) measures the logarithmic frequency with which blocks of length m that are close together remain close together for blocks augmented by one position. Thus, small values of APEN(m) imply strong regularity, or persistence, in a sequence. Alternatively, large values of APEN(m) imply substantial fluctuation, or irregularity" (Pincus and Singer in [88]). In a long truly random sequence, one should expect that APEN(m) $\approx \log 2 \approx 0.6931$.

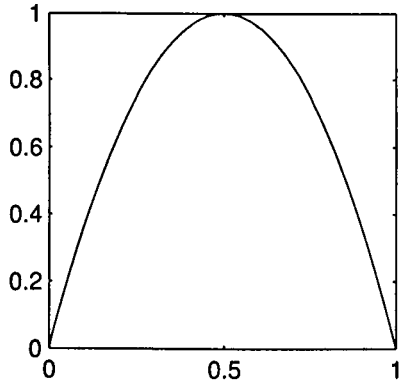
Figure 6.11 depicts the APEN values for five binary sequences taken from:

- the binary expansion of $\pi = 3.1416\ldots$,
- LCG, Linear Congruent Generator, (page 3.6),
- SHA1G, Secure Hash Algorithm Generator described in FIPS 186 (`sha-1`),
- Logistic system (`par`),
- Sine system (`sin`),
- E-Larm, 4 stream combo (`mix`).

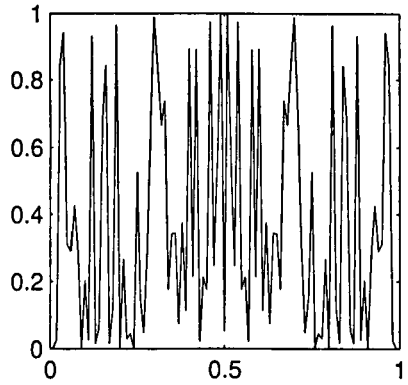
The x-axis reflects the number of bits considered in the sequence. The y-axis reflects the deficit from maximal irregularity, that is, the difference between the $\log 2$ and the observed approximate entropy value. For a fixed sequence length, one can determine which sequence appears to be more random.

For a sequence of 1M bits, (Figure 6.11, upper plot), logistic and sine systems appear more random that LCG. On the lower plot (2M bit sequences) both E-Larm and SHA1G have higher entropy than the binary extension of π. However, for larger block sizes, this is not the case [88].

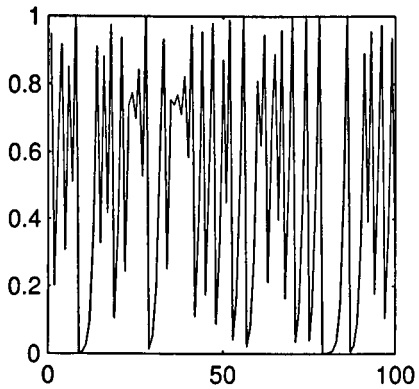
Figure 6.1: Properties of the logistic map



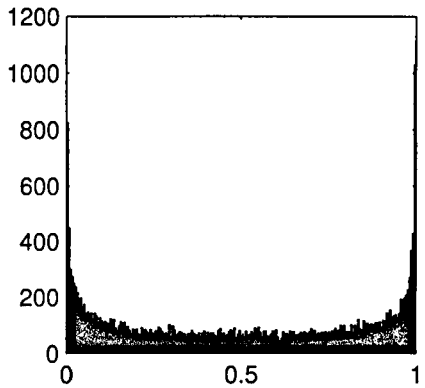
The iterated function $x_{n+1} = f(x_n)$



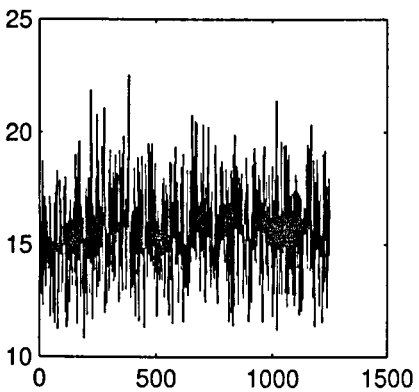
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



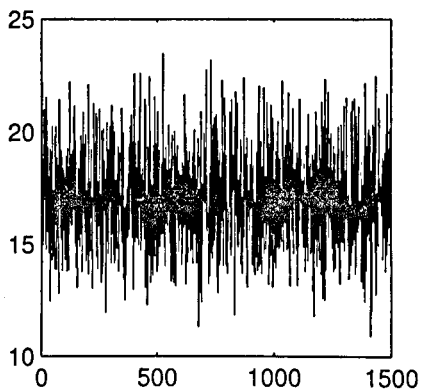
A time series $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments



The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments



The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

6.5 Brief Description of Tests

There are 16 tests in the NIST statistical package, and 12 of them have been applied to analyze pseudo-chaotic sequences. The following is their brief description and parameter values used. The test number and code name in `teletype` is given for further reference.

1 **Frequency (monobit) test** (`frequency`) determines whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.

2 **Frequency test within a block** (`block-frequency`) determines whether the frequency of ones in an M-bit block ($M = 100$) is approximately $M/2$, as would be expected under an assumption of randomness.

3 **Runs test** (`runs`) focuses on the total number of *runs* in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. Among all the runs half should be of length 1, a quarter should be of length 2, an eighth should be of length 3 and so on (as long as the number of runs so indicated exceeds one); for each of these lengths there should be equally many runs of zero bits and runs of one bits.

4 **Test for the longest run of ones in a M-bit block** (`longest-run`) determines whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence.

5 **Binary Matrix Rank Test** (`rank`) focuses on the rank of disjoint submatrices of the entire sequence. The test checks for linear dependence among fixed length substrings of the original sequence.

6 **Spectral Density Test using Discrete Fourier transform** (`fft`) measures peak heights in the Discrete Fourier Transform (DFT) of the sequence. The intention is to detect whether the number of peaks exceeding the 95% threshold is significantly different than 5%.

7 **Non-overlapping Template Matching Test** (`nonperiodic-templates`) measures the number of occurrences of pre-specified target strings. The purpose of this test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. For this test and for the Overlapping Template Matching test, an $m = 10$ bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. If the pattern is found, the window is reset to the bit after the found pattern, and the search resumes.

8 **Overlapping Template Matching Test** (`overlapping-templates`) counts the number of occurrences of prespecified target strings. Both this test and the previous test use an $m = 20$ bit window to search for a specific m-bit pattern. The difference is that when the pattern is found, the window slides only one bit before resuming the search.

9 **Maurer's 'Universal Statistical' Test** (`universal`) focuses on the number of bits between matching patterns (a measure that is related to the length of a compressed sequence). The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random.

10 **Serial Test** (`serial`) is focused on the frequency of all possible overlapping $m = 10$ bit patterns across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern.

11 **Approximate Entropy Test** (`apen`) As with the Serial test, the focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m+1$) against the expected result for a random sequence.

The pseudo-chaotic systems, which have been tested, are given below with their notation code, number of rounds and length of the sequence generated:

Iterated Function	Code	Number of Rounds	Length of Sequence	Fig.
SHA1G	`sha-1`	-	500000	6.12, 6.13
Logistic map	`par`	15	500000	6.12, 6.13
Logistic map	`par-bad`	1	500000	6.12, 6.13
Sine map	`sin`	15	500000	6.12, 6.13
Tangent-feedback map	`tan`	15	500000	6.14, 6.15
Logarithm-feedback map	`log`	15	500000	6.14, 6.15
Sawteeth map	`saw`	15	500000	6.14, 6.15
4 stream combo (E-Larm)	`mix`	16	500000	6.14, 6.15
4 stream combo (E-Larm)	`mix`	16	2000000	6.16

The following test procedure has been applied to the pseudo-chaotic systems:

1. Four different seeds were taken from the valid range (See Tables 5.1 and 5.2 on page 100) using a standard Java PRNG; the control parameter p was assigned a constant as given in Tables 5.1 and 5.2.

2. Four pseudo-chaotic sequence of the length $N = 500,000$ (denoted as T1, T2, T3 and T4) were produced by the E-Larm PRNG.

3. The eleven tests described above were made for each sequence. The test software was compiled on a Linux platform from NIST sources [88].

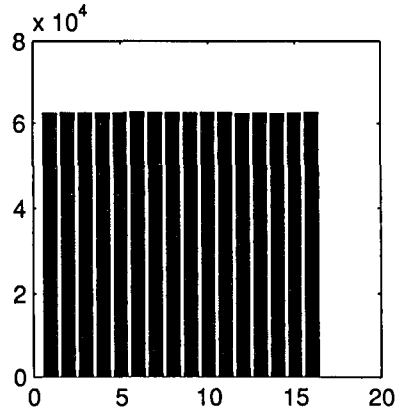
4. (4 × 11) *P-value* were obtained and saved in tables (Figures 6.12-6.16).

5. The average *P-value* was computed from four values.

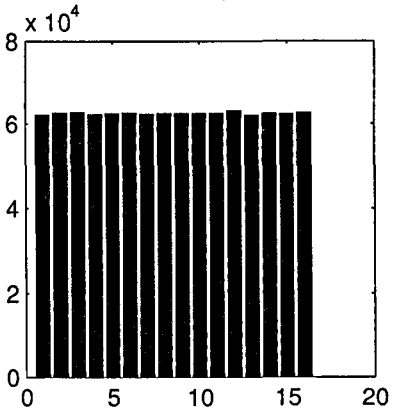
In addition to the standard length $N = 500,000$, four long sequences ($N = 2,000,000$) were generated and tested using the `mix` system (E-Larm). Following is a description of the results given in Figures 6.12-6.16:

`TEST SUMMARY FOR <code>`	`<code>` reference a pseudo-chaotic system (see the table above)
`ALPHA`	Level of significance α
`N`	Length of the sequence
`T1, T2, T3, T4`	`Tn` denotes a *P-value* column corresponding to the `n`-th sequence. Four different sequence were tested
`MEAN`	Average *P-value*
`CONCL`	Test conclusion. The test fails if more then one sequence has *P-value* $< \alpha$
`FINAL CONLUSION`	The generator is considered pseudo-random if it passes all the tests

Figure 6.2: Properties of the logistic map (cont.)

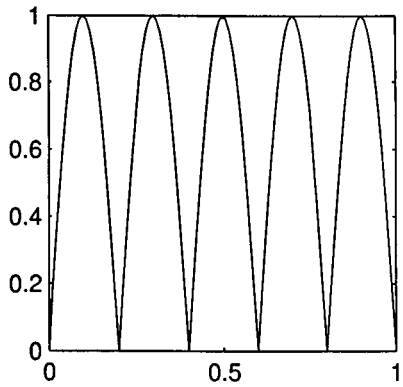


Histogram of 4-bit values obtained using XOR product of the 2 last significant bits ($N = 1,000,000$)

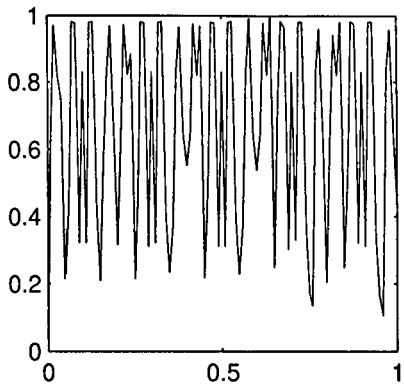


Histogram of 4-bit values obtained using threshold partitioning at $x_{thres} = 0.552$ ($N = 1,000,000$)

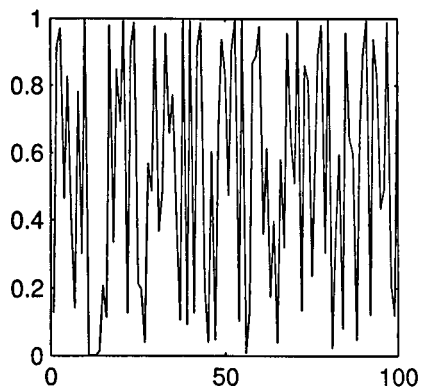
Figure 6.3: Properties of the sine map



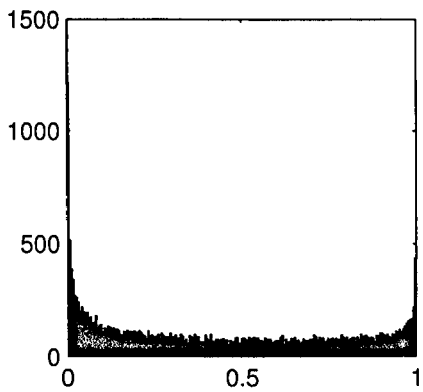
The iterated function $x_{n+1} = f(x_n)$



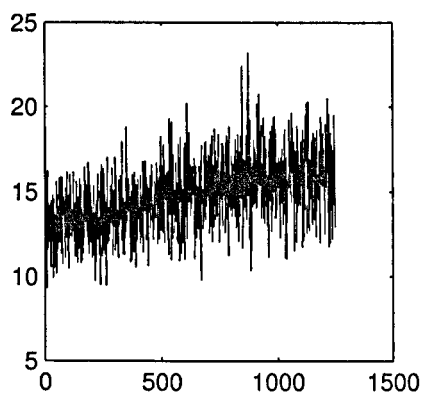
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



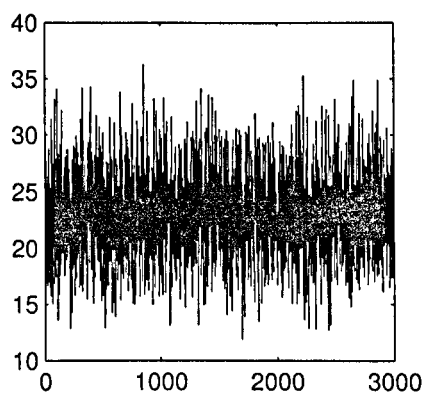
A time series (100 steps) $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments

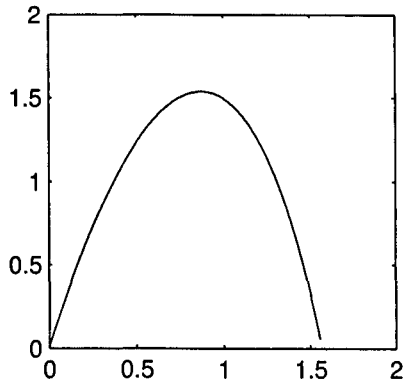


The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments

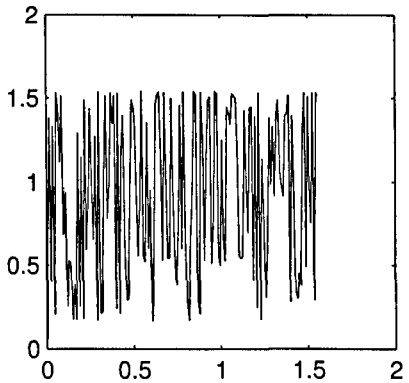


The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

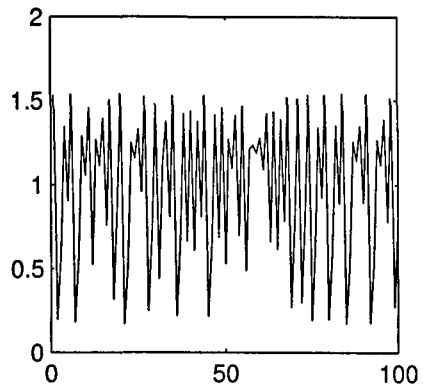
Figure 6.4: Properties of the tangent-feedback map



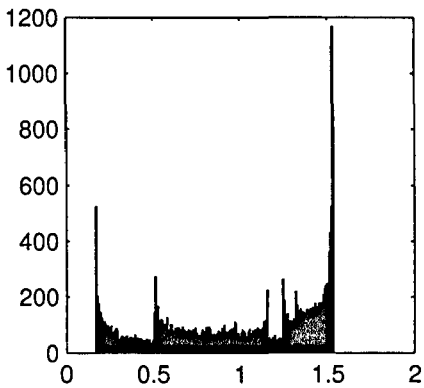
The iterated function $x_{n+1} = f(x_n)$



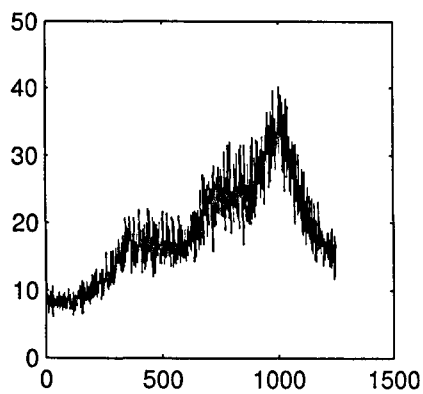
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



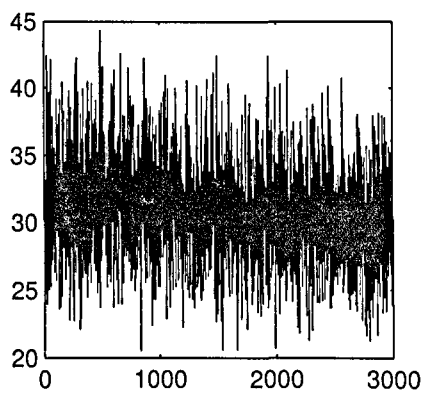
A time series (100 steps) $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments

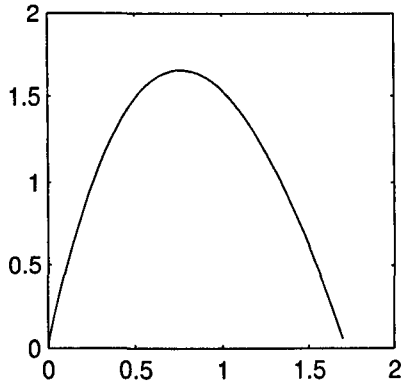


The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments

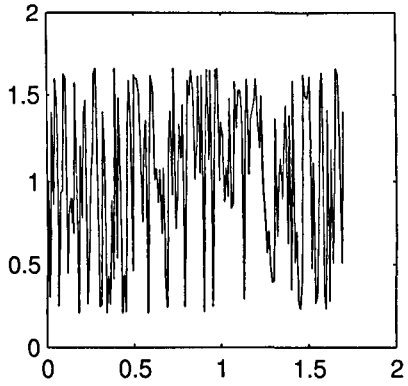


The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

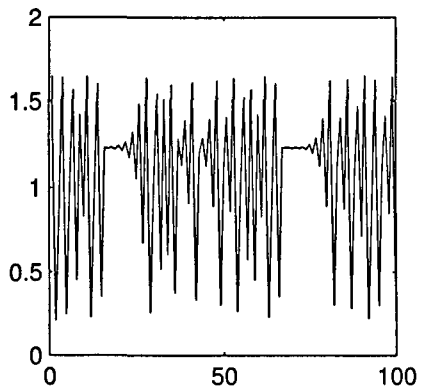
Figure 6.5: Properties of the logarithm-feedback map



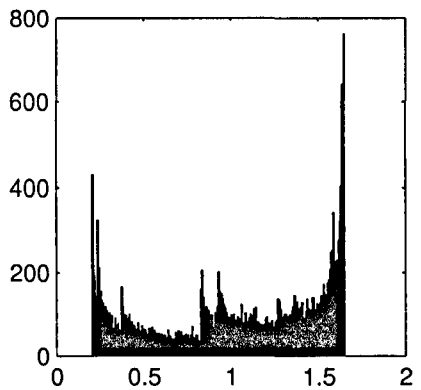
The iterated function $x_{n+1} = f(x_n)$



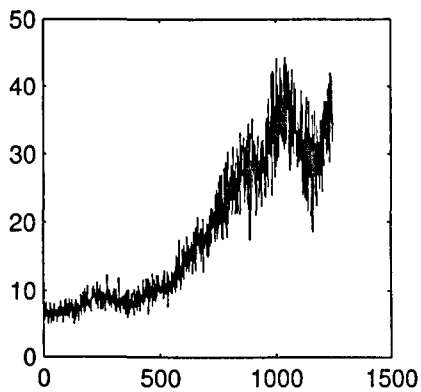
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



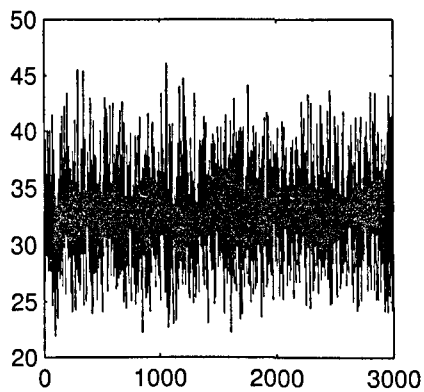
A time series (100 steps) $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments

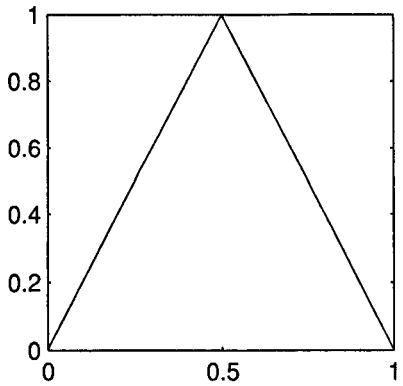


The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments

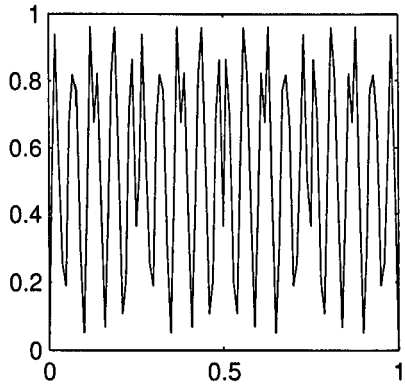


The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

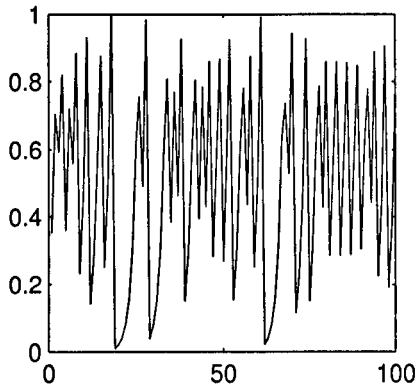
Figure 6.6: Properties of the tent map



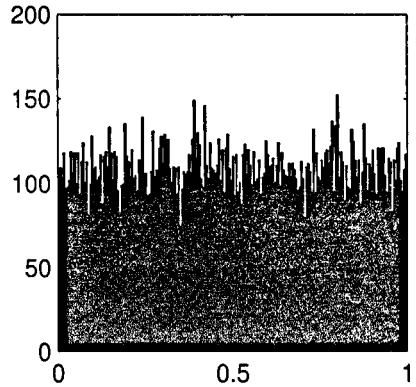
The iterated function $x_{n+1} = f(x_n)$



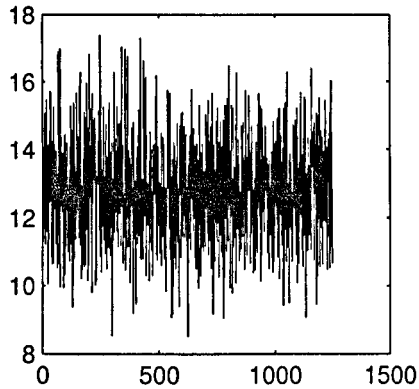
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



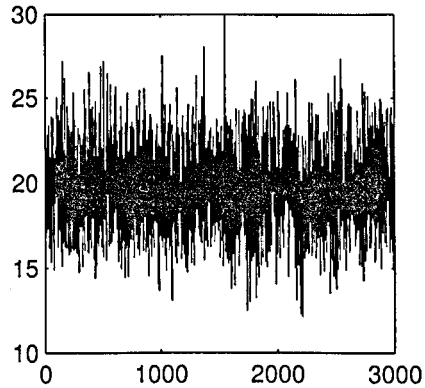
A time series (100 steps) $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments

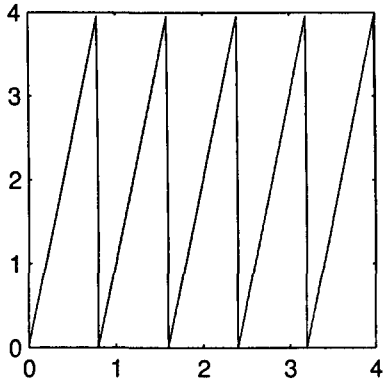


The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments

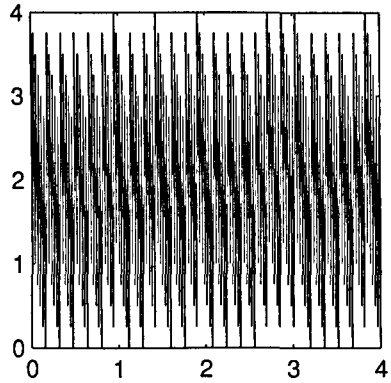


The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

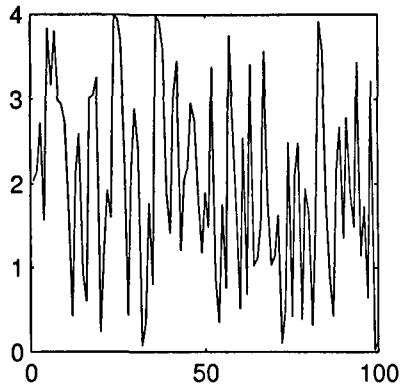
Figure 6.7: Properties of the sawteeth map



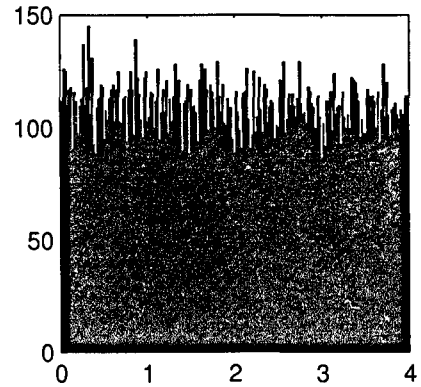
The iterated function $x_{n+1} = f(x_n)$



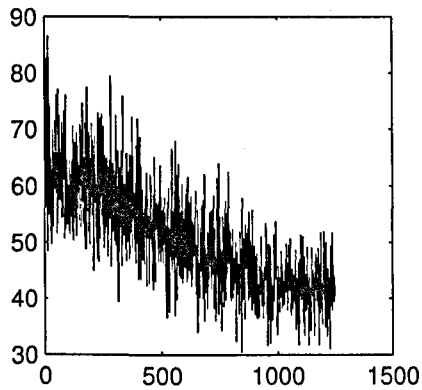
The composition of 15 iterations $x_{n+15} = f^{15}(x_n)$



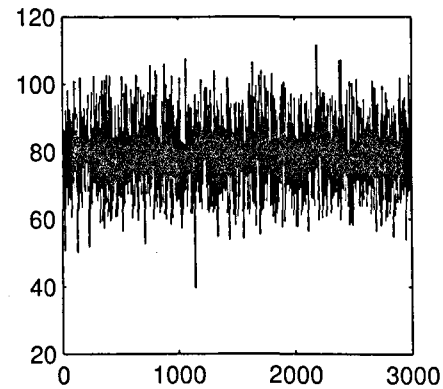
A time series (100 steps) $x_1, x_2, ..., x_N$ (shown $N = 100$)



The probability density function ($N = 2000$) averaged from 10 experiments

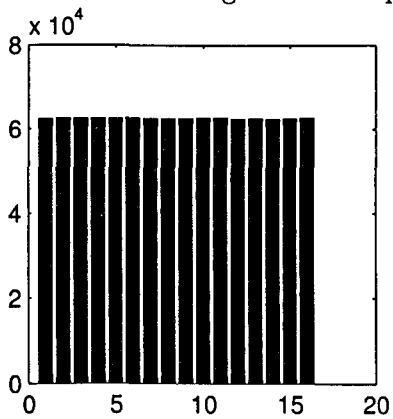


The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ ($N = 6000$) averaged from 10 experiments

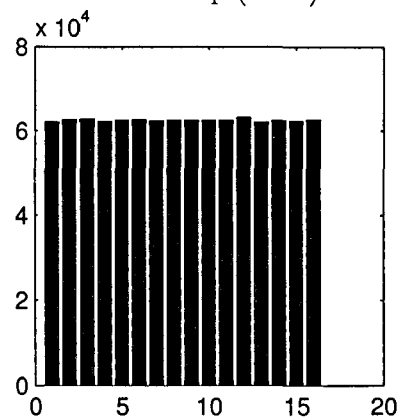


The spectral density Fourier transform of $x_1, x_{15}, \ldots, x_{15N}$ ($N = 6000$) averaged from 10 experiments

Figure 6.8: Properties of the sawteeth map (cont.)

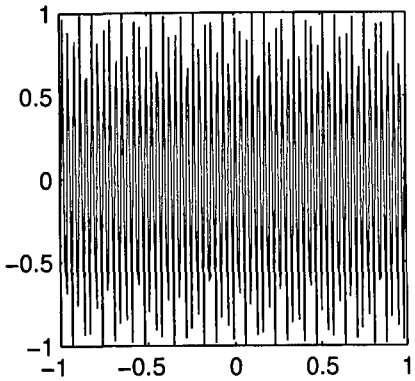


Histogram of 4-bit values obtained using XOR product of the 2 last significant bits

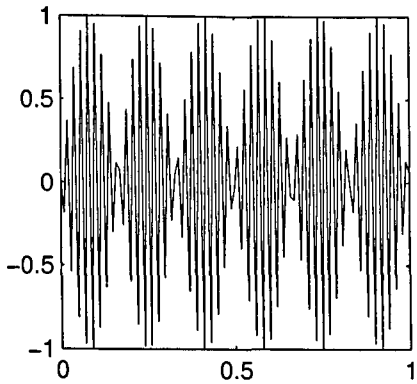


Histogram of 4-bit values obtained using threshold partitioning at $x_{thres} = 2$

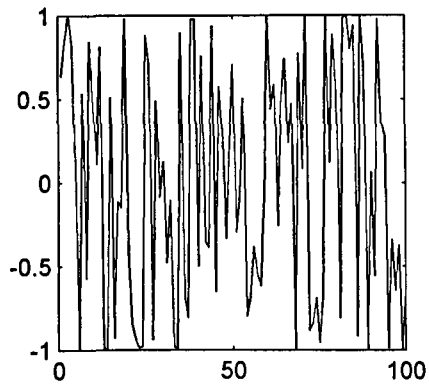
Figure 6.9: Properties of the exactly solvable map



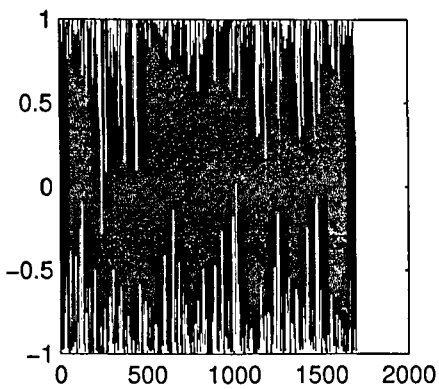
The map $x_{n+1} = f(x_n)$



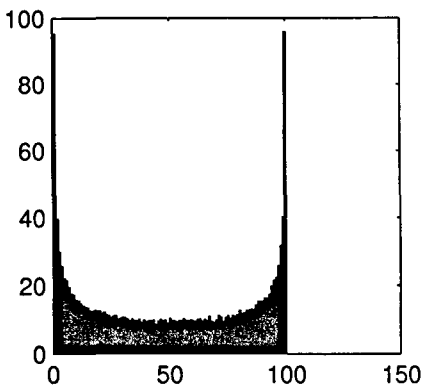
The 15-step map $x_{n+15} = f^{15}(x_n)$



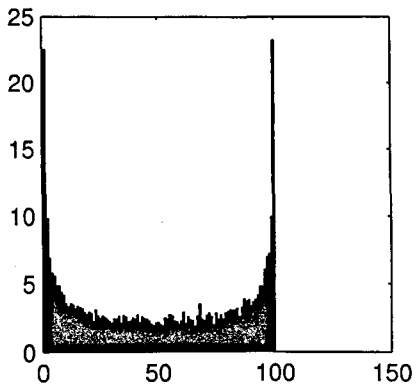
A time series $x_1, x_2, ..., x_N$ (shown $N = 100$)



A time series $x_1, x_2, ..., x_N$ (N=2000 steps) After, in average, $N = 1860$ the variable x becomes Not-a-Number (NaN).

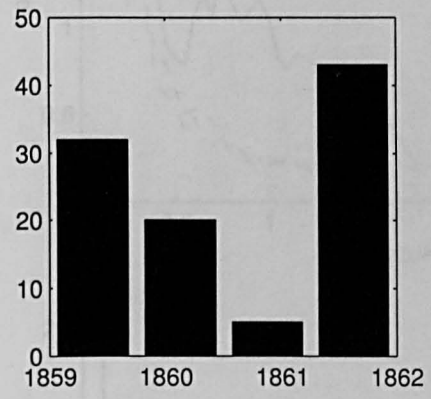


The state distribution ($N = 1000$ first states) averaged from 10 experiments

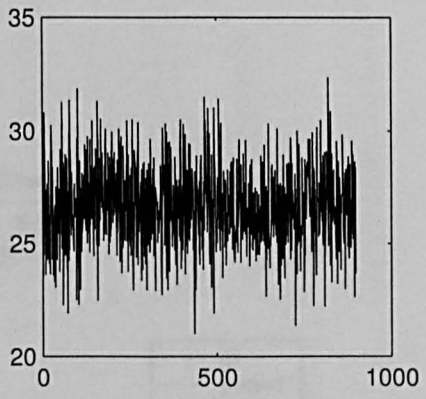


The state distribution ($N = 1000$ last states before NaN) averaged from 10 experiments

Figure 6.10: Properties of the exactly solvable map (cont.)



The distribution of cycle lengths (averaged from 100 experiments)



The spectral density Fourier transform of $x_1, x_2, \ldots, x_N$ $(N = 1859)$ averaged from 10 experiments

Figure 6.11: Approximated entropy plots

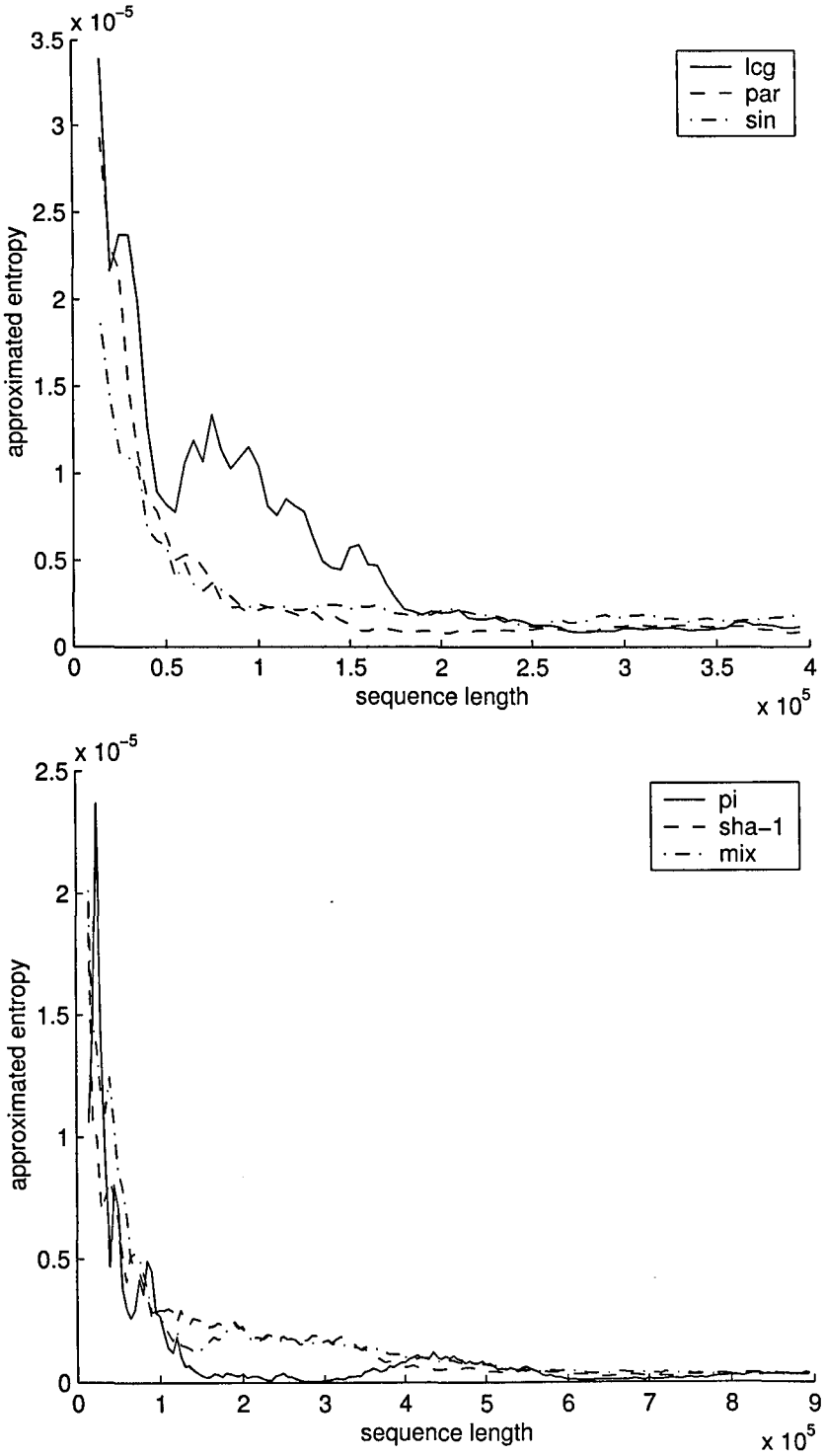


Figure 6.12: Estimation of *P-value* for `sha-1`, `par`, `par-bad`, `sin`

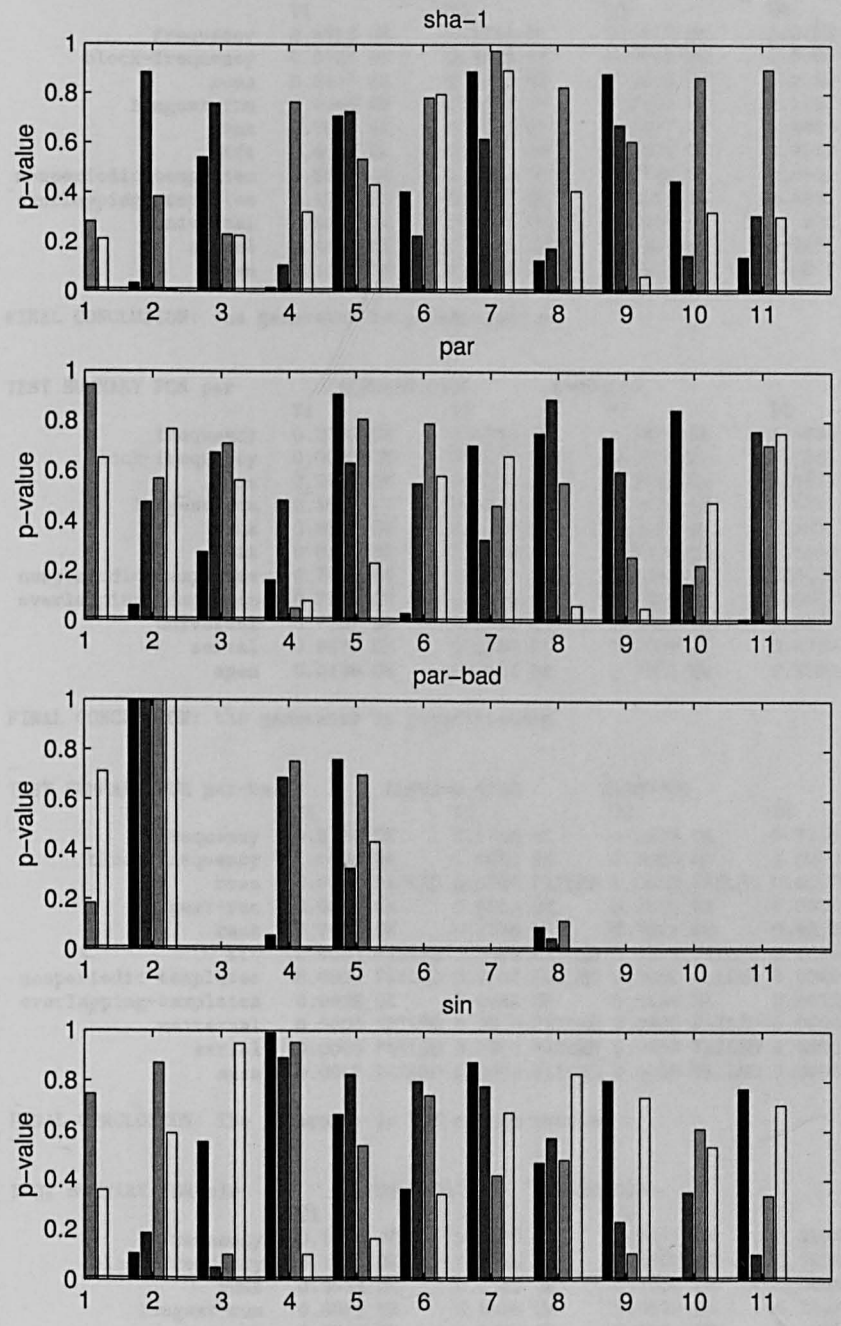


Figure 6.13: Estimation of *P-value* for sha-1, par, par-bad, sin (cont.)

TEST SUMMARY FOR sha-1 ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.4955 OK	0.1414 OK	0.2812 OK	0.2123 OK	0.2826	OK
block-frequency	0.0321 OK	0.8834 OK	0.3829 OK	0.0066 FAILED	0.3263	OK
runs	0.5417 OK	0.7580 OK	0.2309 OK	0.2258 OK	0.4391	OK
longest-run	0.0145 OK	0.1075 OK	0.7653 OK	0.3227 OK	0.3025	OK
rank	0.7088 OK	0.7272 OK	0.5357 OK	0.4314 OK	0.6008	OK
fft	0.4037 OK	0.2258 OK	0.7831 OK	0.7972 OK	0.5524	OK
nonperiodic-templates	0.8883 OK	0.6172 OK	0.9729 OK	0.8935 OK	0.8430	OK
overlapping-templates	0.1275 OK	0.1777 OK	0.8251 OK	0.4092 OK	0.3849	OK
universal	0.8805 OK	0.6734 OK	0.6067 OK	0.0636 OK	0.5560	OK
serial	0.4485 OK	0.1478 OK	0.8656 OK	0.3251 OK	0.4467	OK
apen	0.1425 OK	0.3097 OK	0.8977 OK	0.3057 OK	0.4139	OK

FINAL CONCLUSION: the generator is pseudo-random

TEST SUMMARY FOR par ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.2382 OK	0.0913 OK	0.9436 OK	0.6509 OK	0.4810	OK
block-frequency	0.0593 OK	0.4740 OK	0.5685 OK	0.7666 OK	0.4671	OK
runs	0.2758 OK	0.6723 OK	0.7110 OK	0.5622 OK	0.5553	OK
longest-run	0.1633 OK	0.4838 OK	0.0514 OK	0.0810 OK	0.1949	OK
rank	0.9078 OK	0.6305 OK	0.8071 OK	0.2331 OK	0.6446	OK
fft	0.0318 OK	0.5509 OK	0.7901 OK	0.5819 OK	0.4887	OK
nonperiodic-templates	0.7034 OK	0.3301 OK	0.4643 OK	0.6623 OK	0.5400	OK
overlapping-templates	0.7518 OK	0.8890 OK	0.5553 OK	0.0655 OK	0.5654	OK
universal	0.7357 OK	0.5998 OK	0.2603 OK	0.0552 OK	0.4127	OK
serial	0.8472 OK	0.1530 OK	0.2282 OK	0.4795 OK	0.4270	OK
apen	0.0138 OK	0.7627 OK	0.7051 OK	0.7550 OK	0.5591	OK

FINAL CONCLUSION: the generator is pseudo-random

TEST SUMMARY FOR par-bad ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.8144 OK	0.1746 OK	0.1847 OK	0.7110 OK	0.4712	OK
block-frequency	1.0000 OK	1.0000 OK	1.0000 OK	1.0000 OK	1.0000	OK
runs	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED
longest-run	0.0543 OK	0.6882 OK	0.7521 OK	0.0006 FAILED	0.3738	OK
rank	0.7602 OK	0.3238 OK	0.7001 OK	0.4332 OK	0.5543	OK
fft	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED
nonperiodic-templates	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED
overlapping-templates	0.0936 OK	0.0485 OK	0.1194 OK	0.0012 FAILED	0.0657	OK
universal	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED
serial	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED
apen	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000 FAILED	0.0000	FAILED

FINAL CONCLUSION: The generator is NOT pseudo-random

TEST SUMMARY FOR sin ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.1094 OK	0.9391 OK	0.7471 OK	0.3595 OK	0.5388	OK
block-frequency	0.1032 OK	0.1848 OK	0.8700 OK	0.5903 OK	0.4371	OK
runs	0.5558 OK	0.0510 OK	0.1009 OK	0.9468 OK	0.4136	OK
longest-run	0.9895 OK	0.8905 OK	0.9523 OK	0.1007 OK	0.7332	OK
rank	0.6649 OK	0.8253 OK	0.5374 OK	0.1657 OK	0.5483	OK
fft	0.3636 OK	0.7972 OK	0.7411 OK	0.3446 OK	0.5616	OK
nonperiodic-templates	0.3484 OK	0.8734 OK	0.0687 OK	0.1330 OK	0.3559	OK
overlapping-templates	0.4719 OK	0.5722 OK	0.4818 OK	0.8303 OK	0.5891	OK
universal	0.8035 OK	0.2317 OK	0.1068 OK	0.7345 OK	0.4691	OK
serial	0.0050 FAILED	0.3522 OK	0.6083 OK	0.5354 OK	0.3752	OK
apen	0.9168 OK	0.1010 OK	0.3373 OK	0.7041 OK	0.5148	OK

FINAL CONCLUSION: the generator is pseudo-random

Figure 6.14: Estimation of *P-value* for `tan`, `log`, `saw`, `mix`

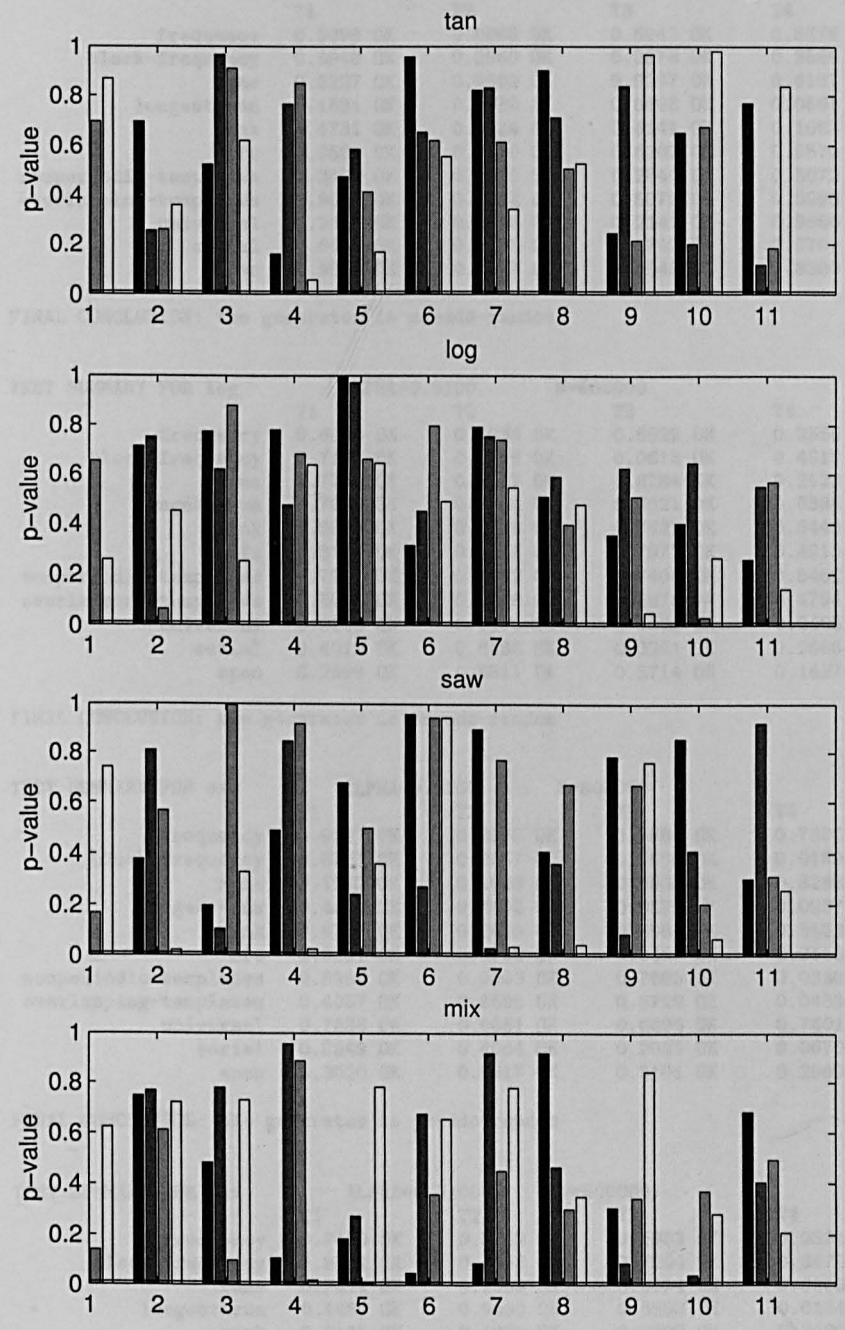


Figure 6.15: Estimation of *P-value* for `tan`, `log`,`saw`,`mix` (cont.)

TEST SUMMARY FOR tan ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.9099 OK	0.6963 OK	0.6942 OK	0.8675 OK	0.7920	OK
block-frequency	0.6948 OK	0.2550 OK	0.2578 OK	0.3586 OK	0.3915	OK
runs	0.5227 OK	0.9660 OK	0.9097 OK	0.6187 OK	0.7543	OK
longest-run	0.1581 OK	0.7639 OK	0.8498 OK	0.0561 OK	0.4570	OK
rank	0.4731 OK	0.5834 OK	0.4141 OK	0.1683 OK	0.4097	OK
fft	0.9561 OK	0.6530 OK	0.6202 OK	0.5570 OK	0.6966	OK
nonperiodic-templates	0.3499 OK	0.6699 OK	0.2840 OK	0.3072 OK	0.4027	OK
overlapping-templates	0.9012 OK	0.7102 OK	0.5079 OK	0.5283 OK	0.6619	OK
universal	0.2447 OK	0.8365 OK	0.2141 OK	0.9650 OK	0.5651	OK
serial	0.6498 OK	0.2003 OK	0.6742 OK	0.9792 OK	0.6259	OK
apen	0.9047 OK	0.1157 OK	0.1848 OK	0.8380 OK	0.5108	OK

FINAL CONCLUSION: the generator is pseudo-random

TEST SUMMARY FOR log ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.6028 OK	0.8034 OK	0.6529 OK	0.2555 OK	0.5787	OK
block-frequency	0.7123 OK	0.7516 OK	0.0613 OK	0.4519 OK	0.4943	OK
runs	0.7748 OK	0.6187 OK	0.8784 OK	0.2528 OK	0.6312	OK
longest-run	0.7805 OK	0.4734 OK	0.6821 OK	0.6384 OK	0.6436	OK
rank	0.9996 OK	0.9716 OK	0.6627 OK	0.6443 OK	0.8195	OK
fft	0.3128 OK	0.4518 OK	0.7972 OK	0.4913 OK	0.5133	OK
nonperiodic-templates	0.7937 OK	0.7542 OK	0.7406 OK	0.5462 OK	0.7087	OK
overlapping-templates	0.5094 OK	0.5908 OK	0.3978 OK	0.4794 OK	0.4944	OK
universal	0.3550 OK	0.6270 OK	0.5082 OK	0.0485 OK	0.3847	OK
serial	0.4012 OK	0.6486 OK	0.0291 OK	0.2666 OK	0.3364	OK
apen	0.2599 OK	0.5511 OK	0.5714 OK	0.1427 OK	0.3813	OK

FINAL CONCLUSION: the generator is pseudo-random

TEST SUMMARY FOR saw ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.9122 OK	0.8875 OK	0.1684 OK	0.7386 OK	0.6767	OK
block-frequency	0.3757 OK	0.8057 OK	0.5660 OK	0.0189 OK	0.4416	OK
runs	0.1952 OK	0.1033 OK	0.9933 OK	0.3263 OK	0.4045	OK
longest-run	0.4881 OK	0.8442 OK	0.9170 OK	0.0237 OK	0.5683	OK
rank	0.6757 OK	0.2403 OK	0.4983 OK	0.3593 OK	0.4434	OK
fft	0.9561 OK	0.2708 OK	0.9415 OK	0.9415 OK	0.7775	OK
nonperiodic-templates	0.8957 OK	0.0303 OK	0.7683 OK	0.0330 OK	0.4318	OK
overlapping-templates	0.4057 OK	0.3595 OK	0.6720 OK	0.0433 OK	0.3701	OK
universal	0.7838 OK	0.0851 OK	0.6696 OK	0.7601 OK	0.5746	OK
serial	0.8549 OK	0.4084 OK	0.2025 OK	0.0670 OK	0.3832	OK
apen	0.3020 OK	0.9217 OK	0.3101 OK	0.2560 OK	0.4475	OK

FINAL CONCLUSION: the generator is pseudo-random

TEST SUMMARY FOR mix ALPHA=0.0100 N=500000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.7450 OK	0.3942 OK	0.1983 OK	0.0394 OK	0.3442	OK
block-frequency	0.9304 OK	0.4553 OK	0.7304 OK	0.8473 OK	0.7408	OK
runs	0.3494 OK	0.1509 OK	0.0394 OK	0.9650 OK	0.3761	OK
longest-run	0.4450 OK	0.9430 OK	0.6398 OK	0.0134 OK	0.5103	OK
rank	0.0195 OK	0.4050 OK	0.0902 OK	0.2480 OK	0.1907	OK
fft	0.6947 OK	0.0394 OK	0.5504 OK	0.7430 OK	0.5068	OK
nonperiodic-templates	0.2308 OK	0.8101 OK	0.0389 OK	0.9889 OK	0.5172	OK
overlapping-templates	0.0385 OK	0.0943 OK	0.8429 OK	0.4590 OK	0.3586	OK
universal	0.9305 OK	0.0503 OK	0.6932 OK	0.4305 OK	0.5261	OK
serial	0.0954 OK	0.4085 OK	0.7430 OK	0.7102 OK	0.4893	OK
apen	0.0245 OK	0.0293 OK	0.9045 OK	0.0893 OK	0.2619	OK

FINAL CONCLUSION: the generator is pseudo-random

Figure 6.16: Estimation of *P-value* for `mix` (long sequence)

TEST SUMMARY FOR mix ALPHA=0.0100 N=2000000

	T1	T2	T3	T4	MEAN	CONCL
frequency	0.0059 FAILED	0.3798 OK	0.1383 OK	0.6236 OK	0.2869	OK
block-frequency	0.7464 OK	0.7665 OK	0.6121 OK	0.7205 OK	0.7114	OK
runs	0.4816 OK	0.7801 OK	0.0974 OK	0.7299 OK	0.5223	OK
longest-run	0.1057 OK	0.9545 OK	0.8864 OK	0.0172 OK	0.4909	OK
rank	0.1834 OK	0.2732 OK	0.0033 FAILED	0.7824 OK	0.3106	OK
fft	0.0467 OK	0.6729 OK	0.3564 OK	0.6530 OK	0.4322	OK
nonperiodic-templates	0.0845 OK	0.9175 OK	0.4479 OK	0.7772 OK	0.5568	OK
overlapping-templates	0.9164 OK	0.4625 OK	0.2981 OK	0.3463 OK	0.5058	OK
universal	0.3020 OK	0.0858 OK	0.3394 OK	0.8423 OK	0.3924	OK
serial	0.0000 FAILED	0.0367 OK	0.3677 OK	0.2799 OK	0.1711	OK
apen	0.6806 OK	0.4030 OK	0.4911 OK	0.2078 OK	0.4456	OK

FINAL CONCLUSION: the generator is pseudo-random

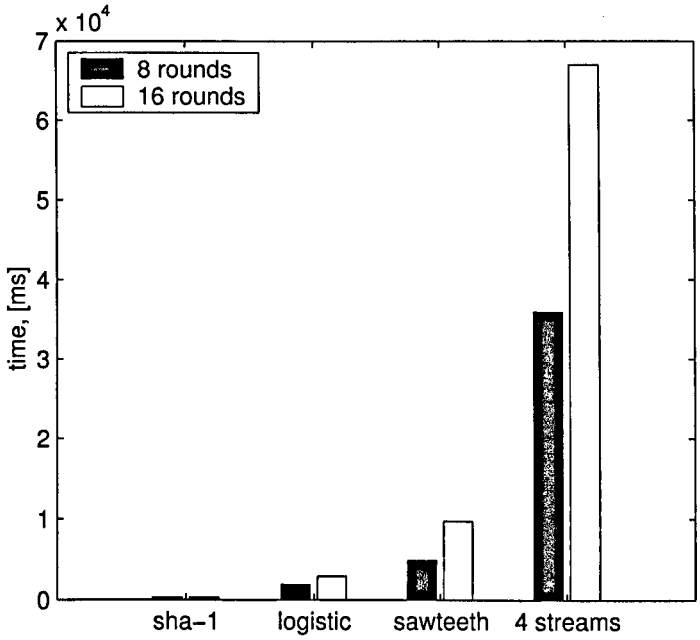


Figure 6.17: Performance evaluation of pseudo-random generators

6.6 Performance Evaluation

Figure 6.17 gives run times in milliseconds of different generators: SHA1G, logistic system, sawteeth system and E-Larm (4 streams combo). The test was made with 500K sequences in Java Runtime Environment version 1.4 running on Pentium III 1 GHz. The time includes the hard-disk output operation. The number of rounds (8 or 16) is the number of intermediate iterations between output bit extracting and mixing. Floating-point generators, particulary, E-Larm, are very slow versus a conventional PRNG (SHA1G). Numerically, the E-Larm algorithm is slower than SHA1G by 100-300 fold depending on pseudo-chaotic systems used.

6.7 Conclusions

The E-Larm implementation of a pseudo-chaotic generator has passed major statistical tests and, from the null hypothesis, can be considered pseudo-random. Long-sequence test ($N = 2,000,000$) for the `mix` has shown more failures of sequence

subtests than the basic tests ($N = 500,000$). Clearly, the statistical tests identify more 'non-random' patterns as we increase the length of sequences. It is very likely that the tests will fail if we take a sufficiently long sequence (say, $N \sim 10,000,000$).

No theoretical cause was found to confirm the pseudo-random property because of iterative rounding, whose crucial impact was too complex to study analytically. Other disadvantages of E-Larm versus conventional PRNG isolate this approach from modern cryptographic applications. In particular:

- The trajectory length is not perfect and not even stable, i.e. the generator has random cycles, including very short patterns. No mathematical instrument is known to estimate the cycle length and find control parameters to produce perfect trajectories, which is not the case for binary generators.

- Compared with modern PRNG's, the E-Larm implementation is complex and inefficient. It requires floating-point processing, float-to-bit transformation and other unnecessary time-consuming operations. A floating-point format is redundant for the present cryptographic application and the overall performance is not acceptable.

Chapter 7

Conclusions

7.1 Theory of Chaos based Cryptography

This thesis has discussed cryptography in the context of chaos theory. Clearly, there is a fundamental relationship between the two sciences. In both chaos theory and cryptography, the object of study is a dynamic system that performs an *iterative nonlinear transformation of information* in an apparently unpredictable but deterministic manner. In terms of chaos theory, the sensitivity to the initial conditions together the mixing property ensures cryptographic confusion and diffusion, as originally suggested by Shannon.

At the same time, there are conceptual differences:

- Chaos theory studies dynamic systems defined on an *infinite state* space (e.g vectors of real numbers or infinite binary strings), whereas cryptography relies on a *finite-state machine* (computer). All chaos models implemented on a computer are approximations, i. e. *pseudo-chaos.*

- Chaos theory studies the *asymptotic behavior* of a nonlinear system ($n \rightarrow \infty$), whereas cryptography focuses on the effect of a *small number of iterations* ($n \ll \infty$).

- Chaos theory is not concerned about the algorithmic complexity of the iterated function, while in cryptography complexity is the key issue, meaning security. In other words, the concepts of *cryptographic security* and *efficiency* have no

counterparts in chaos theory. However, certain mathematical problems from chaos theory can be adopted to cryptography.

- Classical chaotic systems have visually *recognizable attractors*, in which the dimension is fractional. Cryptography attempts to maximize the attractor dimension (to an integer) and to hide any visible structure. Unlike chaos in general, cryptographic systems use all combinations of independent variables to be most unpredictable for an observer.

Chaotic systems are algorithmically random and thus cannot be predicted by a deterministic Turing machine even with infinite power. However, chaotic systems are predictable by a probabilistic Turing machine. Finding probabilistically unpredictable chaotic systems is a central problem for chaos based cryptography.

A rarefied sample $x_k, x_{2k}, \ldots x_{nk}, \ldots$ from a time series $x_1, x_2, x_3, \ldots$, produced by a chaotic and mixing system, is asymptotically independent: for any n, elements $x_{(n-1)k}$ and x_{nk} will be more and more independent as k increases. So far, ergodic theory explains a confusion mechanism of cryptographic systems (i.e. the statistical independence between the ciphertext and the plaintext, or between pseudo-random numbers).

Chaotic systems with analytical solutions of the form $x_n = \Psi(x_0, n)$ and multi-valued map $x_{n+1} = f(x_n)$ can, theoretically, deliver computationally unpredictable (pseudo-random) sequences. The advantage of such a generator is the random access, i.e. any element x_n can be computed directly from the initial condition (seed) x_0. The cryptographic secrecy is kept in the seed and the solution $\Psi(x_0, n)$.

The properties summarized above relate to infinite state systems (true chaos), while cryptographic applications run on finite-state computers. Computer approximations of chaos, called *pseudo-chaos* do not 'converge' with the original model due to the nature of chaos. Therefore, these theoretical results cannot be directly applied to practical cryptography.

In conclusion, we should admit that the impact of chaos theory is quite doubtful. On the one hand the synergetic approach identifies common laws driving cryptographic systems and iterative nonlinear transformation, exponential instability and other properties. On the other hand, this synergetic knowledge is not enough to

design strong cryptographic systems. Neither chaos, nor pseudo-chaos can guarantee pseudo-randomness and resistance to different kinds of cryptanalyses. Chaos theory is not related to number theory and the computational complexity that underpin digital cryptography.

For many years, symmetric ciphers and PRNG's have been deeply studied and do not seem to need serious upgrades. By contrast, asymmetric schemes, which modern security relies upon, require much more attention; they form the basis for such crucial applications as key management and digital signature. However, chaos theory, at least in this project, has not brought any new ideas to asymmetric cryptography.

7.2 Practical Issues

Floating-point arithmetic is the most obvious solution of approximating continuous chaos on a finite-state machine. However, there is no straightforward application to pseudo-random number generation and ciphering. Critical problems are (i) growing rounding-off errors; (ii) the approximated system is not structurally stable, i.e. different initial conditions and parameters yield different cryptographic properties, such as very short orbits, weak plaintexts or weak keys. These problems have been solved in the E-Larm system software (Chapter 5), but in some (insufficient) degree only.

Chaotic systems based on smooth nonlinear functions (e.g. x^2, $\sin(x)$, $\log(x)$) produce sequences with a highly non-uniform distribution. Clearly, these can be predicted by a probabilistic machine. Using the last significant bit (LSB) partitioning and multi-round transformation, we obtain a bit sequence with good statistical properties. It passes all pseudo-randomness tests, from which we conclude that it is pseudo-random.

Many piecewise-linear maps (e.g. sawteeth, tent map) generate sequences, which have theoretically flat distributions. But in practice, these are even more dangerous than nonlinear systems because of the overall effect of linearity, rounding and iterative transformations. Often, histograms of bit sequences produced by piecewise-linear maps, look like a staircase or a large saw. The LSB method does not work (last bits display patterns), instead, the output can be produced from first significant bit(s)

or using a threshold.

As shown in Chapter 6, floating-point chaos based cryptosystems, both nonlinear and piecewise-linear are dramatically inefficient when compared to conventional PRNG's. Other shortcomings, particulary, the lack of theoretical conviction in their pseudo-random property make them unacceptable to modern cryptography.

All conventional cryptographic systems (encryption schemes, pseudo-random generators, hash functions) are *binary pseudo-chaotic systems*, defined on a finite space of strings. Such systems are periodical but have a limited sensitivity to the initial conditions, i.e. the Lyapunov exponents are positive only if measured at the beginning of the process (before one can see the cycles). The mixing property leads to pseudo-randomness.

Pseudo-chaotic systems have typically many orbits of different length. Measuring the minimal, average and maximal length of a system is not a trivial number theory problem. Clearly, ideal cryptographic systems have a single orbit that includes all the possible states.

Iterative block ciphers can be viewed as a combination of two linked pseudo-chaotic systems: data and round-key systems. The iterated function of the data system includes: nonlinear substitutions, row shifts, column mixing etc. The round-key system is a pseudo-random generator providing the sensitive dependence of the ciphertext on the key.

Technically, most of pseudo-random generators are based on the *stretch-and-fold* transformation: first, the state is stretched over a large space (e.g, multiplying or raising in power), then folded into the original state space (using a periodical function such as mod, sin). In mathematical chaos, the stretch-and-fold transformation forms the basis of the majority of iterated functions.

An exact (analytical) solution of a chaotic system has a much stronger stretch-and-fold property. Consequently, it eliminates the precision problem in some degree only: in the beginning, the simulation of chaos is more accurate, but, for long sequences the result is even worse than iterative computations.

Floating-point implementations of analytical solutions do not provide good pseudo-random properties. In particular, orbits can be unpredictably short and their length depends on the initial condition(s).

Another problem is that we know analytical solutions only for certain (boundary) values of the control parameters. This does not allow the use of parameters as a secret key; all the secrecy must be kept in the seed.

7.3 Future Work

As discussed above, chaotic dynamics, which essentially describes real (continuous) state systems, does not really contribute to digital cryptography. However, the following future directions can be suggested with respect to *binary chaos*:

Structurally stable pseudo-chaotic systems It is very important to have a structurally stable cryptosystem, i.e. a system that has (almost) the same cycle length and Lyapunov exponents for all initial conditions and control parameters. Most of the known pseudo-chaotic systems do not possess this quality.

Increasing-precision pseudo-chaotic systems All approximations of chaos studied within this work are based on fixed precision computations. However, it is possible to increase the precision or resolution (e.g. the length of a binary state string) in each iteration. Furthermore, the precision can vary up and down according to a set of rules, defined to estimate the error impact. This technique would be very important in simulation sciences.

Conditions of unpredictability for chaotic systems It is still an open question as to what properties of a chaotic system guarantee its *computational unpredictability.* Known unpredictable systems have emerged from number theory problems. It is possibly that this is the only way to find unpredictable systems.

Invertible pseudo-chaotic systems A one-way transformation forms the basis of a PRNG, whereas a key-dependent invertible transformation is the essence of a cipher or encryption scheme. Most chaos based ciphers extended their chaotic PRNG with invertible transformation (XOR, cyclic shifts and other permutations). The latter transformations can also be considered as pseudo-chaotic maps.

Chaos based asymmetric encryption Modern cryptography is based on asymmetric cryptographic systems, based on *trapdoor functions*, i.e. functions that have the one-way property unless a secret parameter (trapdoor) is known. No counterpart of a trapdoor transformation is known in chaos theory.

Bibliography

[1] С. П. Капица, С. П. Курдюмов, and Малинецкий Г. Г. *Синергетика и прогнозы будущего.* Едиториал УРСС, 2001. // S. P. Kapitsa, S. P. Kurdumov, G. G. Malinensky, *Synergetics and forecasts.*

[2] Ю. А. Данилов and Б. Б. Кадомцев. *Нелинейные волны. Самоорганизация.* Наука, Москва, 1983. `http://www.synergetic.ru/science/index.php?article=dan` // U. A. Danilov, B. B. Kadomtsev, *Nonlinear waves. Self-organization.*

[3] М. В. Капранов and В. Г. Чернобаев. Управляемые генераторы хаотических колебаний на базе систем фазовой синхронизации. *Радиотехнические тетради*, (15), 1998. // M. V. Kapranov, V. G. Chernobaev, *Controlled chaos generators based on phase syncronization systems.*

[4] Г. Г. Малинецкий and А. Б. Потапов. *Современные проблемы нелинейной динамики.* Едиториал УРСС, 2000. // G. G. Malinensky, A. B. Potapov, *Modern problems in nonlinear dynamics.*

[5] М. Гэри and Д. Джонсон. *Вычислительные машины и трудно решаемые задачи.* Мир, Москва, 1982. // M. R. Garey, D. S. Johnson, *Computers and intractability.*

[6] Г. Николис and И. Пригожин. *Познание сложного. Введение.* Мир, 1990. // G. Nicolis, and I. Prigogine, *Exploring complexity. An introduction*, W. H. Freeman and Company, New York.

[7] А. Дмитриев and С. Старков. Новые подходы к решению проблем в системах связи и компьютерных сетях: динамический хаос. *Компьютерра*, (46),

2001. // A. Dmitriev, *Novel solutions in comunication system and computer networks: deterministic chaos.*

[8] П. Р. Халмош. *Теория меры.* ИЛ, 1953. // P. R. Halmos, *Measure theory.*

[9] Я. Г. Синай. *Современные проблемы эргодической теории.* ФИЗМАТЛИТ, Москва, 1995. // Y. G. Sinai, Modern problems in ergodic theory.

[10] В. С. Анищенко. Детерминированный хаос. *Соросовский образовательный журнал*, (6):70–76, 1997. `http://www.nsu.ru/materials/ssl/text/metodics/anishenko.html` // V. S. Anishenko, *Deterministic Chaos.*

[11] Дж. Д. Биркгоф. *Динамические системы.* РХД, Ижевск, 1999. // G. D. Birkhoff, Dynamical Systems.

[12] П. Р. Халмош. *Лекции по эргодической теории.* РХД, Ижевск, 1999. // P. R. Halmos, *Lectures on ergodic theory.*

[13] В. В. Ященко, editor. *Введение в криптографию.* МЦНМО, Москва, 2000. // V. V. Yanchenko, *Introduction to cryptography.*

[14] В. В. Суриков. О термине синергетика. In *Синергетика. Труды семинара.*, volume 3, Москва, 2000. Изд-во МГУ. // V. V. Surikov, *On the term synergetics.*

[15] А. Ю. Лоскутов. Синергетика и нелинейная динамика: новые подходы к старым проблемам. In *Синергетика. Труды семинара.*, volume 3, Москва, 2000. Изд-во МГУ. // A. Y. Loskutov, *Synergetics and nonlinear dynamics: new approaches to old problems.*

[16] Г. Шустер. *Детерминированный хаос. Введение.* Мир, Москва, 1988. // H. G. Schuster, *Deterministic chaos: an introduction*, VCH, Weinheim, 1988.

[17] А. Лоскутов. Нелинейная динамика, теория динамического хаоса и синергетика (перспективы и приложения). *Компьютерра*, (47), 1998. `http://www.cplire.ru/win/InformChaosLab/chaoscomputerra/`

`Loskutov.html` // A. Loskutov, *Nonlinear dynamics, chaos theory and synergetics (perspectives and applications).*

[18] И. Пригожин. *Конец определенности. Время, хаос и новые законы.* РХД, Ижевск, 1999. // I. Prigogine, *The end of certainty. Time, chaos and the new laws of nature*, The Free Press, New York, 1997.

[19] А. Дмитриев. Детерминированный хаос и информационные технологии. *Наука и жизнь*, (5), 2001. `http://nauka.relis.ru/cgi/nauka.pl?07+0105+07105044+HTML` // A. Dmitriev, *Deterministic chaos and information technologies.*

[20] М. Шредер. *Фракталы, хаос, степенные законы. Миниатюры из бесконечного рая.* РХД, Ижевск, 2001. // M. Schroeder, *Fractals, Chaos, Power Laws. Mintutes from an infinite paradise*, W. H. Freeman and Company, New York.

[21] А. Н. Колмогоров. *Теория информации и теория алгоритмов.* Наука, Москва, 1987. // A. N. Kolmogorov, *Information theory and algorithm theory.*

[22] M. S. Baptista. Cryptography with chaos. *Physics Letters A*, 240(1–2):50–54, 1998.

[23] F. L. Bauer. *Decrypted Secrets.* Springer, 1997.

[24] E. Beham. Cryptanalysis of the chaotic-map cryptosystem. In *Proc. of EUROCRYPT'91*, 1991. `http://citeseer.nj.nec.com/175190.html`.

[25] M. E. Bianco and D. Reed. An encryption system based on chaos theory. US Patent No. 5048086, 1991.

[26] J. M. Blackledge. E-larm: super security for e-banking through the application of iterated function systems and fractal modulation for digital encryption. In *Proc. IMA Conference on Fractal Geometry: Mathematical Techniques, Algorithms and Applications*, 2000. not published.

[27] M. Blum and S. Micali. How to generate crytographically strong sequences of pseudo-random bits. *SIAM Journal of Computations*, 13(4):850–864, 1984.

[28] G. Boffetta, M. Cencini, M. Falcioni, and A. Vulpiani. Predictability: a way to characterize complexity, 2001. `http://www.unifr.ch/econophysics/`.

[29] R. Brown and L. O. Chua. Clarifying chaos: examples and counterexamples. *IJBC*, 6(2):219–249, 1996.

[30] A. A. Brudno. Entropy and the complexity of the trajectories of a dynamical system. *Trans. Moscow Mathematical Society*, 44, 1983.

[31] A. B. Cambel. *Applied Chaos Theory. A Paradigm for Complexity.* Academic Press Inc., New-York, 1993.

[32] L. Cappelletti. An FPGA implementation of a chaotic encryption algorithm. Master's thesis, Universita Degli Studi di Padova, 2000. `http://www.lcappelletti.f2s.com/Didattica/thesis.pdf`.

[33] J. M. Carroll, J. Verhagen, and P. T. Wong. Chaos in cryptography: the escape from the strange attractor. *Cryptologia*, 16(1):52–72, 1992.

[34] B. V. Chirikov and F. Vivaldi. An algorithmic view of pseudorandomness. *Physica D*, (129), 1999.

[35] Y. H. Chu and S. Chang. Dynamic cryptography based on synchronized chaotic systems. *Electronic Letters*, 35(12), 1999.

[36] Y. H. Chu and S. Chang. Dynamic data encryption system based on synchronized chaotic systems. *Electronic Letters*, 35(4), 1999.

[37] F. Dachselt, K. Kelber, and W. Schwarz. Chaotic coding and cryptoanalysis. `http://citeseer.nj.nec.com/355232.html`.

[38] F. Dachselt, K. Kelber, J. Vandewalle, and W. Schwarz. Chaotic versus classical stream ciphers – a comparative study, 1998. `citeseer.nj.nec.com/article/dachselt98chaotic.html`.

[39] Whitfield Diffie and Martin Hellman. New directions in cryptography. 2(6):644–654, 1976.

[40] W. Ebeling, L. Molgedey, J. Kurths, and U. Schwarz. Entropy, complexity, predictability and data analysis of time series and letter sequences, 1999. `http://citeseer.nj.nec.com/395066.html`.

[41] T. Erber, T. Rynne, W. Darsow, and M. Frank. The simulation of random processes on digital computers: unavoidable order. *Journal of Computational Physics*, (49):349–419, 1983.

[42] M. J. Feigenbaum. The universal metric properties of nonlinear transformations. *J. Stat. Physics*, (21):669–706, 1979.

[43] A. Fog. Chaotic random number generators, 1999. `http://www.agner.org/random/theory/`.

[44] J. Fridrich. Discrete-time dynamical systems under observational uncertainty. *J. Appl. Math. Comp.*, (83):181–207, 1993.

[45] J. Fridrich. Symmetric ciphers based on two dimension chaotic map. *IJBC*, 8(6):1259–1284, 1998.

[46] J. Fridrich and J. Geer. Reconstruction of chaotic orbits under finite resolution. *J. Appl. Math. Comp.*, (80):129–159, 1995.

[47] R. Buckminster Fuller and E. J. Applewhite. *Synergetics.Explorations in the Geometry of Thinking.* Macmillan Publishing, New York, 1979.

[48] P. Gacs. Lecture notes on descriptional complexity and randomness. Computer Science Department, Boston University, 2001. `http://cs-pub.bu.edu/faculty/gacs/Home.html`.

[49] J. B. Gallagher and J. Goldstein. Sensitive dependence cryptography, 1996. `http://www.navigo.com/sdc/`.

[50] O. Goldreich. Introduction to complexity theory. Lecture Note, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel, 1999.

[51] J. A. González and R. Pino. Chaotic and stochastic functions. *Physica*, 276A:425–440, 2000.

[52] H. Gutowitz. Cryptography with dynamical systems. ESPCI, Laboratoire d'Electronique, Paris, France, 1995. `http://www.santafe.edu/~hag/crypto/crypto.html`.

[53] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori. A secret key cryptosystem by iterating chaotic map. In *Proc. of EUROCRYPT'91*, pages 127–140, 1991. `http://link.springer.de/link/service/series/0558/bibs/0547/05470127.htm`.

[54] H. Haken. *Synergetics, an introduction: nonequilibrium phase transitions and self-organization in physics, chemistry, and biology, 3rd rev.* Springer-Verlag, New York, 1983.

[55] B.-L. Hao. *Elementary symbolic dynamics and chaos in dissipative systems.* World Scientific Pub Co, 1989.

[56] J. Hastad. Pseudo-random generators under uniform assumptions. In *Proceedings 22nd Annu. ACM Symp. on Theory of Computing*, pages 385–404, 1990.

[57] M. K. Ho. Chaotic encryption techniques. Master's thesis, Department of Electronic Engineering, City University of Hong Kong, 2001. `http://personal.cityu.edu.hk/~50115849/ces/`.

[58] S. Hollasch. IEEE standard 754: floating point numbers, 1998. `http://research.microsoft.com/~hollasch/cgindex/coding/ieeefloat.html`.

[59] J. Hosack. The use of Chebysev mixing to generate pseudo-random numbers. *Journal of Computational Physics*, (67):482–486, 1986.

[60] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proc. 21st Annu. ACM Symp. on Theory of Computing*, pages 230–235, 1989.

[61] E. A. Jackson. Perspectives in nonlinear dynamics. Cambridge University Press, 1991.

[62] S. Katsura and W. Fukuda. Exactly solvable models showing chaotic bahavior. *Physica*, (130A):597–605, 1985.

[63] M. P. Kennedy, R. Rovatti, and G. Setti. *Chaotic electronics in telecommunications*. CRC Press, 2001.

[64] D. Knuth. *The art of computer programming - seminumerical algorithms*, volume 2. 2nd ed. Addison-Wesley: Reading, Massachusetts, 1981.

[65] L. Kocarev. Chaos and cryptography, 2001. `http://rfic.ucsd.edu/chaos/ws2001/kocarev.pdf`.

[66] L. Kocarev. Chaos-based cryptography: a brief overview. *Circuits and systems*, 1(3):6–21, 2001.

[67] L. J. Kocarev, K. S. Halle, K. Eckert, and L. O. Chua. Experimental demonstration of secure communications via chaotic synchronization. *IJBC*, 2(3):709–713, 1992.

[68] Z. Kotulski, J. Szczepańs, K. Gyrski, A. Gyrska, and A. Paszkiewicz. On constructive approach to chaotic pseudorandom number generators. In *Proc. of the Regional Conference on Military Communication and Information Systems*, volume 1, pages 191–203. CIS Solutions for an Enlarged NATO, RCMIS, 2000. `http://www.ippt.gov.pl/~zkotulsk/CPRBG.pdf`.

[69] Z. Kotulski and J. Szczepański. Discrete chaotic cryptography. new method for secure communication. In *Proc. NEEDS'97*, 1997. `http://www.ippt.gov.pl/~zkotulsk/kreta.pdf`.

[70] Z. Kotulski and J. Zczepanski. On the application of discrete chaotic dynamical systems to cryptography. DCC method. *Biuletyn Wat Rok*, XLVIII(10):111–123, 1999. `http://www.ippt.gov.pl/~zkotulsk/wat.pdf`.

[71] D. Lai, G. Chen, and M. Hasler. Distribution of the Lyapunov exponent of the chaotic skew tent map. *IJBC*, 9(10):2059–2067, 1999.

[72] L. A. Levin. One-way function and pseudorandom generators. *Combinatorica*, 7(5):357–363, 1987.

[73] L. Lovász. Computation complexity. Lecture Notes. `http://ftp.cs.yale.edu/pub/lovasz.pub/`.

[74] George Marsaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995. `http://stat.fsu.edu/pub/diehard/`.

[75] N. Masuda and K. Aihara. Finite state chaotic encryption system, 2000. `http://www.aihara.co.jp/rdteam/fs-ces/`.

[76] R. Matthews. On the derivation of a chaotic encryption algorithm. *Cryptologia*, (13):29–42, 1989.

[77] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptology*. CRC Press, 1996. `http://www.cacr.math.uwaterloo.ca/hac/`.

[78] V. I. Oseledec. A multiplicative ergodic theorem: Lyapunov charcteristic numbers for dynamical systems. *Trans. Mosc. Math. Soc.*, 19(197), 1968.

[79] E. Ott, editor. *Chaos in Dynamical Systems*. Cambridge University Press, 1993.

[80] N. Paar. Robust encryption of data by using nonlinear systems, 1999. `http://www.physik.tu-muenchen.de/~npaar/encript.html`.

[81] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chans and fractals. New frontiers of science*. Springer, 1992.

[82] V. A. Protopopescu, R. T. Santoro, and J. S. Tolliver. Fast and secure encryption-decryption method. US Patent No. 5479513, 1995.

[83] Nikolai V. Ptitsyn, Jonathan M. Blackledge, and Valery M. Chernenkiy. Deterministic chaos in digital cryptography. In *Proceedings of IMA Conference on Fractal Geometry*. Horwood Publishing, 2002.

[84] V. Rijmen and J. Daemen. Rijndael algorithm specification, 1999. `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/`.

[85] T. Ritter. Ciphers by ritter, 1991. `http://www.ciphersbyritter.com/`.

[86] T. Ritter. The efficient generation of cryptographic confusion sequences. *Cryptologia*, (15):81–139, 1991. `http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM`.

[87] R. L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, (2):120–126, 1978.

[88] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. NIST, 2001. `http://csrc.nist.gov/rng/rng2.html`.

[89] J. Scharinger. Secure and fast encryption using chaotic Kolmogorov flows. Johannes Kepler University, Department of System Theory, 1998. `http://www.cast.uni-linz.ac.at/Department/Publications/Pubs1998/Scharinger98f.htm`.

[90] B. Schneier. *Applied cryptography, second edition.* John Wiley & Sons, Inc, 1996. `ISBN 0-471-12845-7`, `http://www.ssl.stu.neva.ru/psw/crypto/appl_rus/appl_cryp.htm`.

[91] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(4):379–423, 623–526, 1948.

[92] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[93] M. I. Sobhy and A. E. D. Schehata. Secure e-mail and databases using chaotic encryption. *Electronic Letters*, 36(10), 2000.

[94] M. Svensson and J. E. Malmquist. A simple secure communications system utilizing chaotic functions to control the encryption and decryption of messages. Project report for the course 'Chaos in science and technology',

Lund Institute of Technology, Dept. of physics, Subdept. of mathematical physics, 1996. `http://www.efd.lth.se/~d92ms/chaoscrypt.html`.

[95] J. Szczepański and Z. Kotulski. Pseudorandom number generators based on chaotic dynamical systems. *Open Systems & Information Dynamics*, 8(2):137–146, 2001. `http://www.ippt.gov.pl/~zkotulsk/open.pdf`.

[96] A. Tsonis, editor. *Chaos: from theory to applications.* Plenum Press, New York, 1992.

[97] H. Waelbroeck and F. Zertuche. Discrete chaos, 1998. `http://papaya.nuclecu.unam.mx/~nncp/chaos98.ps`.

[98] J. Walker. A pseudorandom number sequence test program, 1998. `http://www.fourmilab.ch/random/`.

[99] D. D. Wheeler. Problems with chaotic cryptosystems. *Cryptologia*, (12):243–250, 1989.

[100] D. D. Wheeler. Supercomputer investigations of a chaotic encryption algorithm. *Cryptologia*, (15):140–150, 1991.

[101] H. White. Algorithmic complexity of points in dynamical systems. *Ergodicity Theory Dynamical Systems*, 13, 1993.

[102] S. Wolfram. Random sequence generation by cellular automata. *Advances in Applied Mathematics*, (7), 1986.

[103] W. K. Wong. Chaotic encryption technique. City University of Hong Kong, Department of Electronic Engineering, Hong Kong, 1999. `http://kitson.netfirms.com/chaos/`.

[104] C. J. Wu and Y. C. Lee. Observer-based method for secure communication of chaotic systems. *Electronic Letters*, 36(22), 1999.

[105] A. C. Yao. Theory of applications of trapdoor functions. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

[106] Philip R. Zimmerman. The international PGP home page. `http://www.pgpi.org/pgpi/`.

Index

AES, 29
analytical solution, 75, 82
attack, 34
attractor, 37, 43
attractors, 37

bifurcation, 40

cellular automata, 86
chaos, 36
    binary, 41, 84
    continuos, 36
chaos theory, 19
chaotic map
    Baker map, 87
    cellular automata, 86
    Chebyshev map, 71
    discrete tent map, 86
    logistic map, 72
    Matthews map, 76
    multi-valued map, 82
    RANROT, 85
    sawteeth map, 62
    tent map, 76
chaotic system
    binary, 41, 84
    real space, 36
cipher, 26
    asymmetric, 27
    block, 29
    stream, 29
    symmetric, 27
    transposition, 28
    Vernam, 31
ciphertext, 26
class BPP, 49
class NP, 49
class P, 49
complexity, 46, 47
    algorithmic, 50
    symbolic, 51
    trajectory, 55
compromise, 33
confusion, 32
cryptanalysis, 17, 33
cryptography, 15
cryptology, 17

decryption, 26
DES, 29
diffusion, 32

E-Larm, 91
encryption, 26
encryption scheme
    see cipher, 26

entropy, 47
    approximated, 109
    conditional, 52
    Kolmogorov-Sinai, 54
    KS, 39
    Shannon, 52
    symbolic trajectory, 54
ergodicity, 40

fractal, 37
function
    hash, 58
    one-way, 57

Hamming distance, 41
hard-core predicate, 58

incompressibility, 46, 50
independence
    asymptotic, 62
information, 24
information security, 15
initial condition, 25, 35

Kerchhoffs principle, 33
key, 26
Kolmogorov flows, 87

language, 49
level of significance, 104
LFSR, 85
Lyapunov exponents, 39
    for binary pseudo-chaos, 85

Markov sequence, 53

one-time pad, 31
orbit, 25, 35

p-value, 105
partitioning, 54, 92
    LSB, 92
    threshold, 93
PDF, 52
PGP, 27
plaintext, 24
PRNG, 32, 59
probability distribution function, 52
probability ensemble, 56
pseudo-chaos, 65
pseudo-random generator, 32, 59
    chaos-based, 61
    multi-stream, 79
pseudo-randomness, 46, 57

randomness
    algorithmic, 46, 47, 50
    asymptotic, 62
    tests, 102
    true, 45, 52
RSA, 27

session, 97
state space, 37
substitution, 28
symbolic dynamics, 54
synergetics, 17
system
    chaotic, 36
    cryptographic, 25

dynamic, continuous, 35
ergodic, 40
measure preserving, 40
mixing, 41

trajectory, 25, 35
ideal, 67
symbolic, 54
transposition, 28
Turing machine, 48
deterministic, 49
nondeterministic, 49
probabilistic, 49

unpredictability
absolute, 45
computational, 46

white noise, 45